18-12-2017

UNIT - I ⇒ ASSIGNMENT - I

① Explain about Paradigm of Programming language.

A1 (Paradigms of programming language என்பது programmer எப்படி program எழுத வேண்டும் என்ற model-ஐ குறிக்கிறது. மேலும் இது program execute ஆகும் போது அதன் view குறிக்கிறது) A1

Types:

i) unstructured programming
ii) Procedure programming
iii) structured programming
iv) object oriented programming (OOPS)

unstructured programming:

* Program code ஆனது ஒரே block-ஆக காணப்படும்.
* Program-ல் தேவை கண்டுபிடிப்பது மற்றும் திருத்துவது கடினம்.
* global data மற்றும் goto statement காணப்படும்.

eg: BASIC

Procedure programming:

* பெரிய problem-ஆனது எந்த எண்ணிக்கை-யான சிறிய problem-களாக divide செய்யப்படும் இது function என்றும் (or) Procedure என்றும் அழைக்கப்படும்.

* இது top-down programming technique-ஐ பயன்படுத்துகிறது.

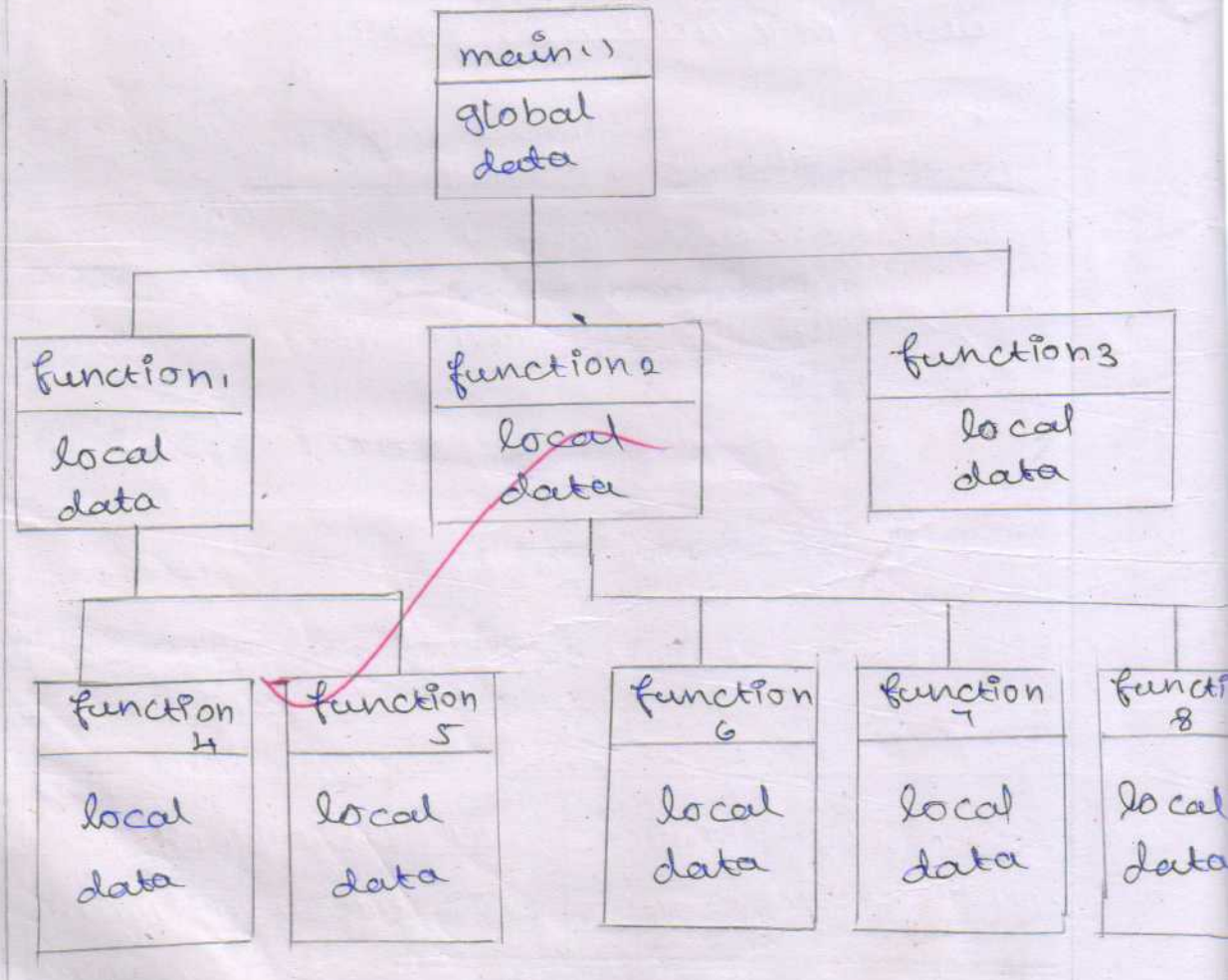
* data ஒரு function-லிருந்து இன்னொரு function-கு easy-ஆக எளிய முடியும்.

* global மற்றும் local variable பயன்படுத்துகிறது.

* data hiding technique அதிக அளவில் பயன்படுத்தப்படுவது.

* புதிய function-ன் data-ஐ add செய்வது மிகவும் கடினம்.

eg: C

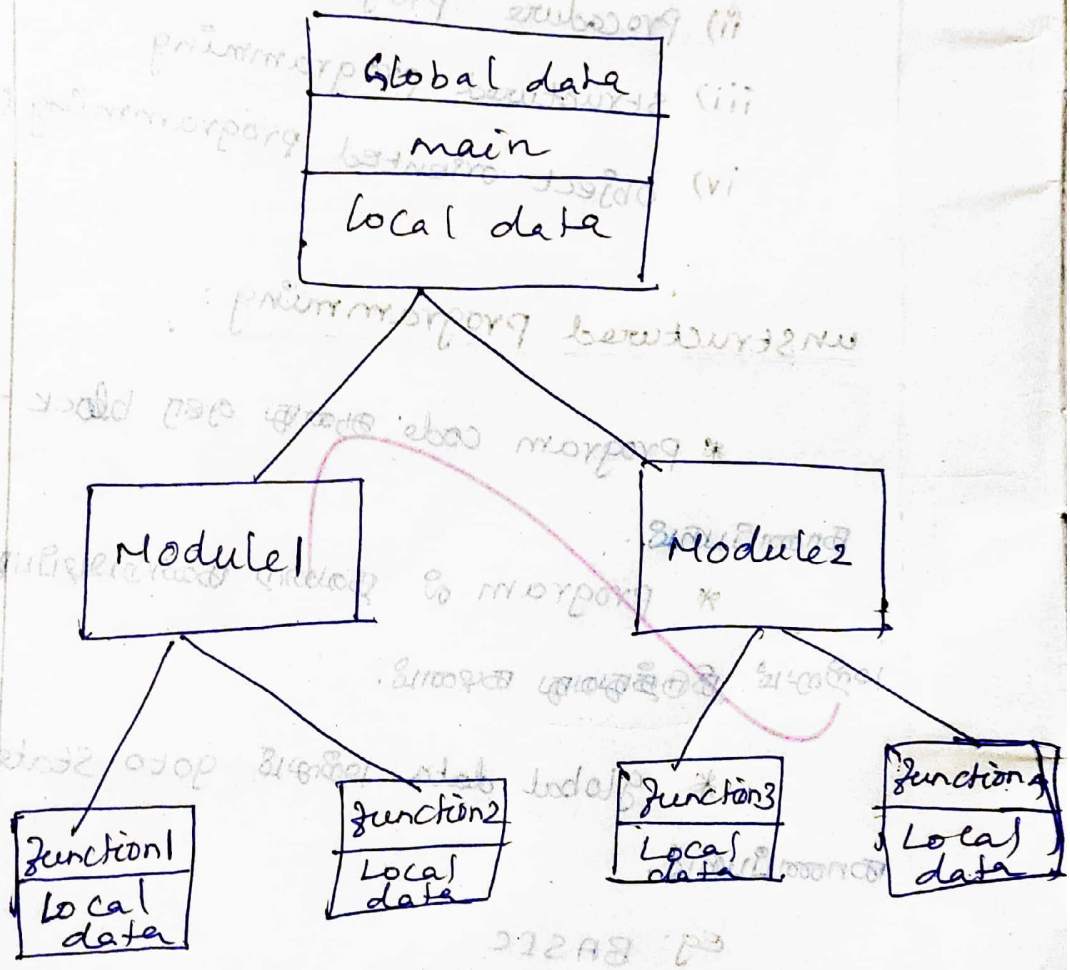


3. Structured Programming

* ಇದರಲ್ಲಿ Problem ಅನ್ನು modules ಆಗಿ divide ಮಾಡಲಾಗುತ್ತದೆ.

* ಇದು Procedure programming ನ subset ಆಗಿದೆ.

* ಇದರಲ್ಲಿ goto statement ಇರುವುದಿಲ್ಲ.



eg: C++

* Object oriented programming

* Problem divide into objects

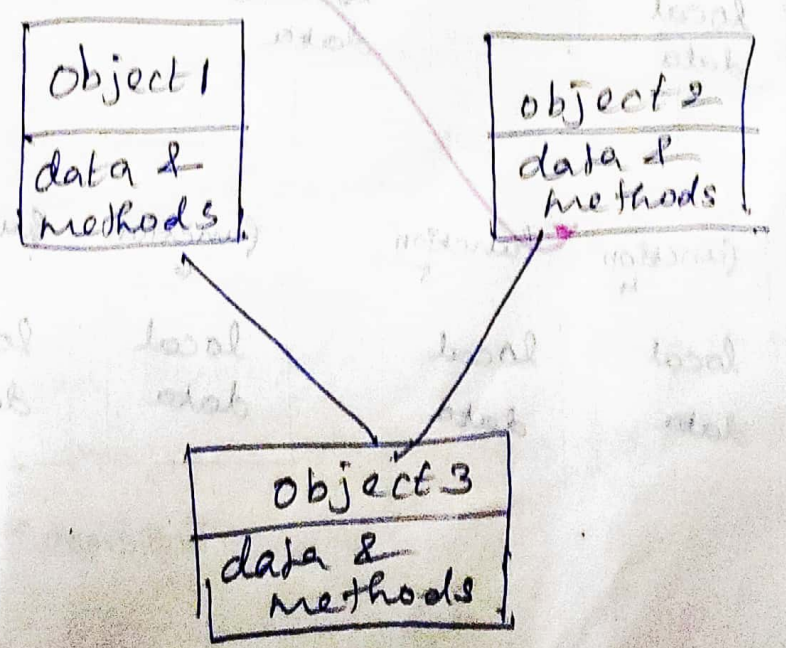
* Data hiding easy to access

* Bottom up programming technique

* Data hiding technique

* New data types

eg: java



3. data hiding செய்ய இயலாது.	data hiding செய்யலாம்.
4. data-வை easy-ஆக access செய்யலாம்.	data-வை easy-ஆக access செய்ய முடியாது.
5. இந்த function மூலம் data-வை அழிப்பது கடினம்.	இந்த function மூலம் data-வை அழிப்பது எளிது.

2. Explain the features of java.

1) simple, small, familiar:

Java ஒரு சிறிய language. இதன் தேம் program எழுதுவது எளிதானது. ஏன் எனில், இதில் pointer, header files, operator overloading and virtual base class போன்ற concept-கள் கிடையாது.

2) object oriented:

Java ஒரு முழுமையான object oriented language ஆகும். இதில் உள்ள எல்லா program-களும் class-இனை அடிப்படையாகக் கொண்டது.

3) Compiled and Interpreted:

B3

மனிதனால் computer language-ல்

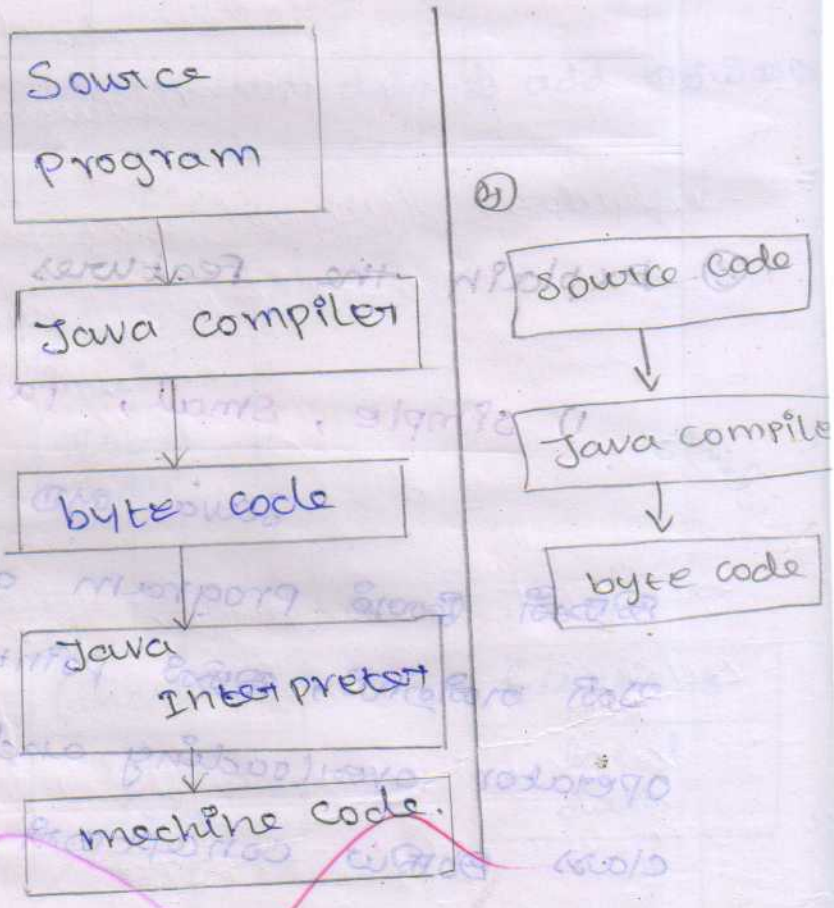
Compiler (அல்லது) Interpreter என்பேன். அனால்

Java Compiler மற்றும் Interpreted சேர்ந்த

கலவையாகும்.

Java என்பது ஒரு இருநிலை அமைப்

பாகும்.



Java compiler, Java programm

byte code-ஆக மாற்றுகிறது. Java Interpreter

byte code-னை machine code-ஆக

மாற்றுகிறது. B3

4) Architecture neutral or platform ~~interpreter~~ independent:

Java compiler platform சாராத byte code-ஐ உருவாக்குகிறது. இந்த byte code-ஐ எந்த வகையான System-திலும் செயல்படுத்தலாம்.

diagram

5) Portable:

Java compiler உருவாக்கிய byte code-ஐ ஒரு System-திலிருந்து மற்றொரு System-திற்கு எளிதாக மாற்றி செயல்படுத்தலாம்.

6) High speed:

Java interpreter byte code-ஐ பயன்படுத்துவதால் இதனுடைய வேகம் அதிகமாகிறது.

7) multithreaded and interactive:

Java-வில் multithreaded வசதியின் மூலம் ஒன்றுக்கு மேற்பட்ட செயல்களை ஒரு நேரத்தில் செயல்படுத்தலாம். Java-ஐ பயன்படுத்தி interactive program-களை உருவாக்க முடியும்.

8) Dynamic and extensible:

C, C++ function-களை Java Program-ல் பயன்படுத்தலாம். இந்த function-களுக்கு தேவையான header file-களை runtime-ல் சேர்த்து கொள்ளலாம். Java program-களை எளிதாக மாற்றிக் கொள்ளலாம்.

9) Distributed:

Java எனப்படுவது networking வசதிகளை கொண்டது எனவே Java மூலம் network application-களை உருவாக்கலாம். இந்த வசதியில் நமக்கு வலைதளத்தின் பலவேறு பகுதியின் உள்ளவர்களை ஒரே application-ல் ஒன்றிணைந்து செயலாற்றலாம்

10) Robust:

Garbage collection-ஐ பயன்படுத்தி memory-களை நிர்வகிக்கிறது.

Exception handling technique-ஐ பயன்படுத்தி அசாதாரணமான சூழல் ஏற்படாத வண்ணம் தடுக்கிறது.

11) Secured:

Internet-லிருந்து வரும் Java code-களை interpret செய்வதற்கு முன்னால், அதில் virus உள்ளதா என்று சோதனை செய்கிறது. Virus இல்லை என்றால் மட்டும் அந்த program-களை interpret செய்கிறது. Digital singing method-ஐ பயன்படுத்தி நம்முடைய code-ஐ secure செய்ய முடியும்

---

4. Explain the basic concept of OOPS?

Ans: (i) Object

ii) Class

iii) Data abstraction

iv) Data encapsulation

v) Inheritance

vi) Polymorphism

vii) Dynamic binding

viii) message passing -) Bi

1) **Object:**

A2 (Object என்பது ஒரு பொருள், இது data மற்றும் அதன் மேல் செயல்படும் function-களை தொகுத்து வைத்திருப்பது. Function என்பது data-வை operate செய்யப்படுகிறது.

Objects			Data (Attributes)	functions
Rollno	NAME	Class	Number of columns Number of Rows	Insert Delete search
1001	RAM	BA		
2001	Somu	MA		

A2

2) **Class:**

class என்பது ஒரு மாதிரியான object-களின் தொகுப்பு ஆகும். class-ல் function-ஐ கீழ்க்கண்டவாறு define செய்ய வேண்டும்.

```
class classname
    variable declaration
    :
    method definition
    :
end class
```

eg:-

```
class student
{
    int regno;
    char name[20];
    int total;
    char branch[10];
    void read()
    {
      ....
      ....
    }
};
```

3) Data abstraction:-

A3 (தேவையான details-களின் தொகுப்பு abstraction ஆகும். இது தேவையான detail-ஐ தவிர மற்றவற்றை மறைக்கிறது.

Data abstraction என்பது பயனர்பற்றி data-களின் தொகுப்பு, இது class-ன் data object ஆக்குகிறது.) A3

Data encapsulation:-

B1 (Data encapsulation என்பது ஒரு object-லிருந்து மற்றொரு object-க்கு உடைய தகவலை பாதுகாக்க உதவும் technique ஆகும்.

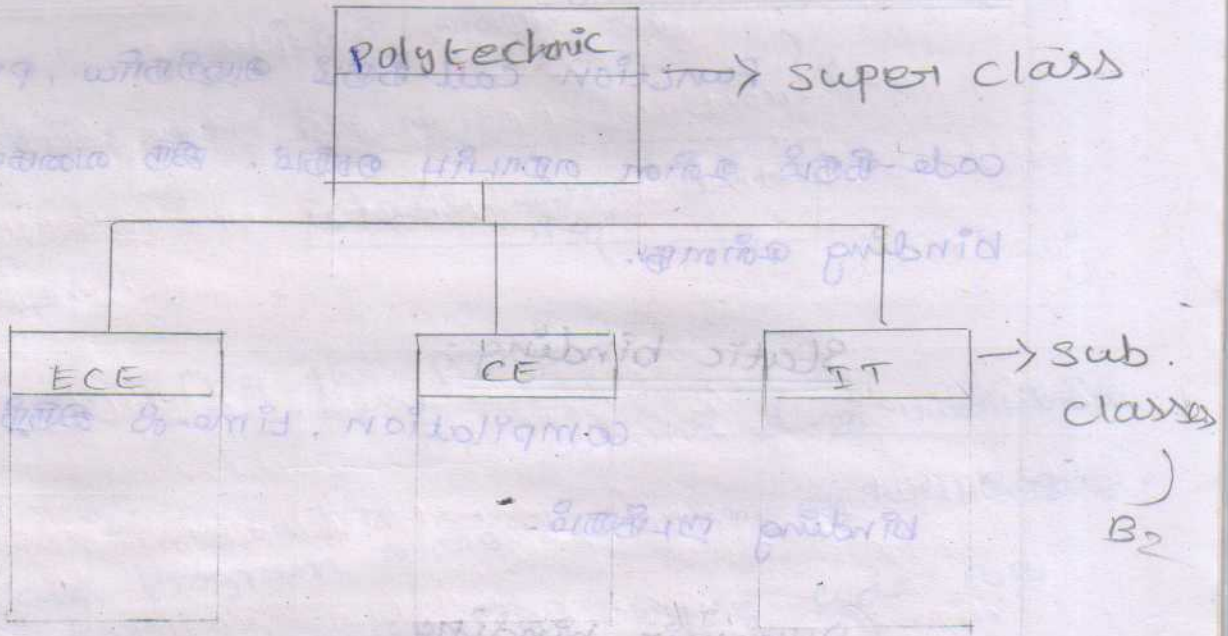
இது concept அனைத்து data hiding என்பதாகும்.) B1

5) B2

Inheritance:

Inheritance என்பது ஒருகிற்கு class லிருந்து ஒரு புதிய class-ஐ derive செய்யும் ஒரு process ஆகும்.

Derive செய்யப்படுபவனிற்கும் class base class என்பதும். Derive செய்யப்பட்ட class derived என்பதும்

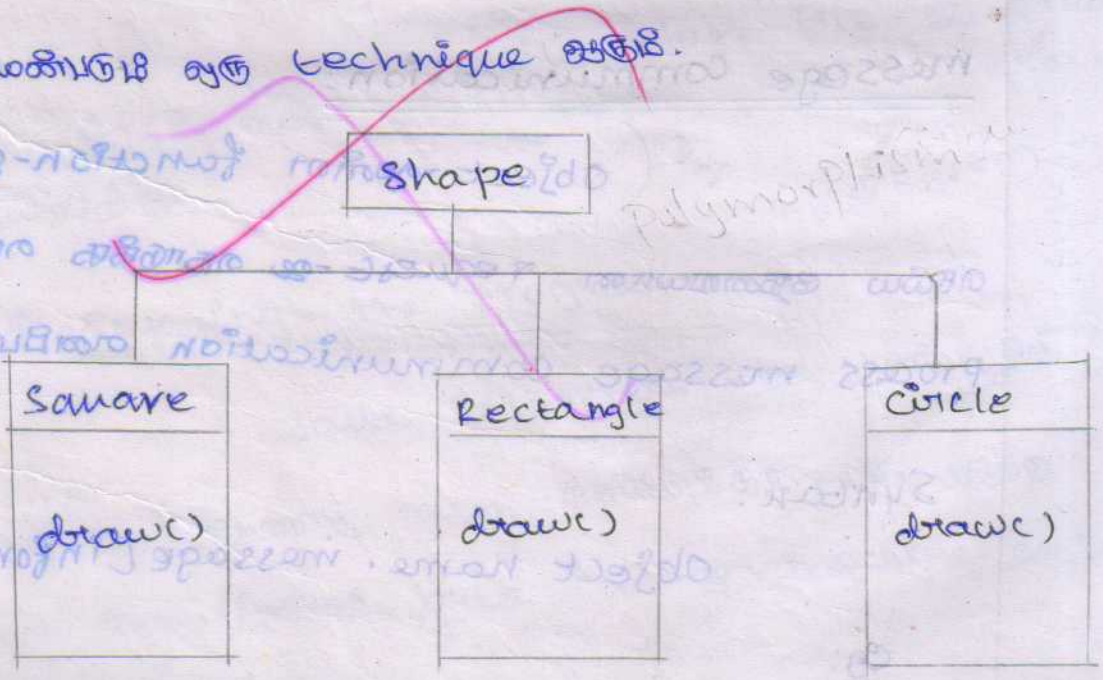


B2

6)

Polymorphism:

Polymorphism என்பது ஒன்றுக்கு மேற்பட்ட function, definition-ஐ function name-ல் ஒரே பயன்படுத்தும் ஒரு technique ஆகும்.



மேலே உள்ள example-ல் function draw() எல்லா Class-யிலும் define செய்யப்பட்டுள்ளது. ஆனால் function draw()-ன் operation, ஒவ்வொரு class-லும் வித்தியாசமாக இருக்கும்.

1) **Dynamic binding:**

Function call-க்கும் அதற்குரிய, program code-க்கும் உள்ள தொடர்பு. இரு வகையான binding உள்ளது.

Static binding:

compilation, time-ல் நிகழ்ந்து binding முடிக்கும்.

Dynamic binding:

Run time-ல் நிகழ்ந்த binding முடிக்கும் இதனை late binding என்றும் அழைக்கலாம்.

2) **message communication:**

object-ன் function-ஐ execute செய்ய தேவையான request-ஐ கொடுக்க செய்யும் process message communication எனப்படும்.

Syntax:

Object name. message (information);

Eg:

S1.read();

5. Explain how will you create and execute a Java program with example.

① Creating the program:

EDIT, WORD, NOTEPAD என்ற ஏதாவது ஒரு Software-ஐ பயன்படுத்தி program-ஐ type செய்ய வேண்டும், அதை Java directory-ல் store செய்ய வேண்டும்.

```
class name .java
```

② Compiling the program:

Java-ன் command-ஐ பயன்படுத்தி Java program அதை compile செய்யப்படுகிறது. இது Java program-ஐ byte code ஆக மாற்றுகிறது.

```
javac classname .java
```

Java compile class file-ஐ create செய்கிறது.

③ Running the program:

Java interpreter-ஐ பயன்படுத்தி program அதை run செய்யப்படுகிறது. இந்த Java Interpreter byte code-ஐ machine code ஆக மாற்றுகிறது

```
java classname.
```

Exam:

```
class Sum
{
 public static void main(String args[]).
 {
  int a = 40;
  int b = 30;
  int c = a+b;
  System.out.println("sum = "+c);
  }
}
```

* இந்த program-ஐ Sum.java என்று save செய்ய வேண்டும்.

* compile செய்ய javac Sum.java

* run செய்ய java Sum

---

6. Explain about JDK.

JDK என்பது Java program-ஐ ru
செய்வதற்கான tools-களை கொண்டுள்ளது.

i) appletviewer ⇒ Java applet-ஐ run
செய்ய பயன்படுகிறது.

ii) javac - Java compiler, இது Java Progr
-ஐ byte code ஆக மாற்றுகிறது.

iii) iii) java - java interpreter, இது java byte code-ஐ machine code-ஆக மாற்றுகிறது.

iv) javap - java disassembler, இது byte code file-ஐ program description-ஆக மாற்றுகிறது.

v) Javadoc - Java source code file-லிருந்து HTML format document-ஆக create செய்கிறது.

vi) jdb - Java debugger, இது program-லுள்ள error-ஐ கண்டுபிடிக்க பயன்படுகிறது.

vii) javah - C, C++ போன்ற native method-ஐ use செய்யும் போது header file-ஐ create செய்கிறது.

---

7. Explain about API (Application Programming Interface)

API - என்பது application program எழுதுவதற்கு தேவையான package-களை கொண்டுள்ளது

i) Language Support Package:

இது basic java features-ஐ implement செய்வதற்கு தேவைப்படும் classes மற்றும் methods-களின் தொகுப்பு.

ii) utilities Package:

Date மற்றும் time போன்ற utility class-களின் தொகுப்பு

iii) Applet Package :-

Applet-ஐ Create செய்வதற்கு தேவைப்படும் class-களின் தொகுப்பு

iv) awt package :-

Graphical user interface-ஐ Implement செய்வதற்கு தேவைப்படும் class-களின் தொகுப்பு.

v) Input/Output Package :-

Input, output function-க்கு தேவையான class-களின் தொகுப்பு.

---

⑧ Explain about java tokens.

95 (token என்பது java-வில் ஒரு தனியான element ஆகும். ஒவ்வொரு token-க்கு இடையில் white space இருக்க வேண்டும். white space என்பது blank space, tap, & Carriage Return ஒரு இருக்கலாம்.) All

Types:

1) Keyword
2) Identifer
3) Literal
4) operator
5) Seperator.) 85

Keyword:

A 14/ Keyword என்பது Java language-ல் ஏற்கனவே word-கள் ஆகும். keyword-களுக்கு ஒரு standard meaning-ஐ கொண்டு இருக்கும். இந்த meaning-ஐ யாரும் மாற்ற முடியாது. keyword கண்டிப்பாக lowercase-ல் letter-ல் எழுத வேண்டும்.

eg. int, float, if, for) A4

Identifier:

B6 Identifier என்பது ஒரு name. இது Program-ல் வரும் variable, class, array etc. ஆகியவற்றிற்கு name கொடுக்க பயன்படுகிறது.

Rules:

* இதில் alphabets, digit, dollar sign, மற்றும் underscore வரலாம்.
* முதல் எழுத்து கண்டிப்பாக alphabet ஆக இருக்க வேண்டும்.
* Identifier எந்த நீளத்திலும் இருக்கலாம்.
* Case sensitive கொண்டது.
* keywords-ஐ கூடாது.

eg:

Sum, total.) B6

~~Literal~~ Literal

Literal என்பது Constant-ஐ குறிக்கும்

Types:

* Integer Literal
* float Literal
* character literal
* String literal
* Boolean Literal.

ஏதேனும் literal-ஆக எந்த Store செய்தாலும் value-வின் வகை-ஐ வைத்திருக்கிறது.

Operator:

operator என்பது ஒரு symbol அது data மீது செயல்படும் operation-ஐ குறிக்க பயன்படுகிறது.

eg: +, -, >, <, *.

Separator:

Separator என்பது Java language-ல் Special symbol ஆகும். இவைகள் Java code எங்கே divide செய்ய படுகின்றது என்பதை குறிக்கும்

Symbol	Usage
;	⇒ Statement-ஐ முடிக்க பயன்படுகிறது

, ⇒ Variable - கண்டறிய பயன்படுகிறது

. ⇒ Package - கண்ட பிரிக்க பயன்படுகிறது.

( ) ⇒ argument - கண்ட enclose செய்ய பயன்படுகிறது.

{ } ⇒ ஒரு block of code - ஐ enclose செய்ய பயன்படுகிறது.

[ ] ⇒ array - யைக் பயன்படுகிறது. AIC

---

9. Explain command line arguments

command line மூலமாக program - கு கொடுக்கப்படும் parameter - கள் ஆகும். இவை program run செய்யப்படும் போது கொடுக்கப்படும்.

Syntax:

```
C> java classname arguments to be passed
```

java - key word

classname - class - பெயர்

arguments to be passed - list of arguments blank space - ஐ separate செய்யப்படும்.

main method-ன் syntax:

```
Public static void main (String args
```

String args[] என்பது String object-ன் array. இது command line வழியாக வழங்கப்படும் argument-ஐ array-ல் store செய்யும்.

eg:

```
Class Command
{
Public static void main (Strin
                         args[])
{
for (int i=0; i< args.length ;
System.out.println (args[i]);
}
}
```

C> java Command CA CNS

args[0] - CA

args[1] - CNS

args[2] = DS

2.1.2018

UNIT - II

ASSIGNMENT - II

① Explain about operator:

operator என்பது symbol ஆகும் data மீது செயல்படும் operation - ஐ குறிக்கும்.

Types:

1. Arithmetic operator.
2. Logical operator.
3. Relational operator
4. Shorthand assignment operator.
5. Increment & decrement operator
6. Conditional operator
7. Bitwise operator
8. Special operator.

1. Arithmetic operator:

Arithmetic operation - ஐ செயல்படுத்த பயன்படுகிறது.

Binary operator:

Symbol	operation
+	Addition
-	Subtraction
*	multiplication
/	Division
%	Modulo operator.

Unary operator:

Symbol	Operation
-	unary minus
++	increment
--	decrement.

2. Relational operator:

இரண்டு operand-களை compare செய்வதற்கு relational operator பயன்படுகிறது.

operator1 or operator2

Symbol	operation.
<	less than
<=	less then or equal to
>	greater than
>=	greater than or equal to
==	equal to
!=	not equal to

3. Logical operator:

இரண்டு relational expression-களை compare செய்வதற்கு logical operator பயன்படுகிறது

operand to operand2

Operator	Operation
&&	AND
||	OR
!	NOT

x	y	x&&y	x || y	!x
0	0	0	0	1
0	1	0	1	1
1	0	0	1	0
1	1	1	1	0

4. Short hand assignment operator:

இந்த operators assignment statment-ஐ simplify பண்ணுவதற்கு பயன்படுகிறது.

+=, -=, *=, /=, %=

eg: x = x+10;
x += 10;

5. Increment and decrement operator.

i) increment operator

Variable உடன் 1ஐ add பண்ணுவதற்கு இந்த operator பயன்படுகிறது.

++ Variable (Prefix) or Variable ++ (Postfix)

eg:

++x

ii) Decrement operator.

இந்த operator variable உடன் 1ஐ Subtract செய்வதற்கு பயன்படுகிறது.

-- variable or variable --

eg:

x --

6. Bitwise operator.

Bit-களில் operation -ஐ செயல்படுத்த பயன்படுகிறது.

operator	Operation
&	Bitwise AND
|	Bitwise OR
^	Bitwise Ex-OR
>>	Bitwise right shift
<<	Bitwise left shift
~	Bitwise complement.

7. conditional operator:

expression1? expresspon2 : expression3

இதில் மூன்று operand-களை எடுப்பதால் ternary operator என்று அழைக்கப்படுகிறது.

முதல் expression 1 என்ன மதிப்பை கணக்கிடுகிறது. True -ஆக இருந்தால் expression 2 evaluate செய்யும். false ஆக இருந்தால் expression

eg:

```
x = a>b ? a : b
```

9. Special operator:

இதில் முக்கியமான Special operator-கள் இரண்டு.

i) instanceof operator
ii) dot operator (.)

instanceof operator:

instanceof operator என்பது ஒரு object ஆனது ஒரு குறிப்பிட்ட class-ல் உள்ளதா, இல்லையா என கண்டறிய பயன்படுகிறது.

```
objectname instanceof classname
```

object ஆனது ஒரு class-ஐ சார்ந்திருந்தால் true value-ஐ return செய்கிறது. இல்லையெனில் false value-ஐ return செய்கிறது.

eg: ramu instanceof sport

dot operator (.)

dot operator (.) class-ல் காணப்படும் method மற்றும் variable-ஐ access செய்ய பயன்படுகிறது.

```
objectname . variable or method
```

eg:

```
ramu . add()
```

Sem 2. Explain about Decision making and branching statements.

break & continue

5 Types of Statements

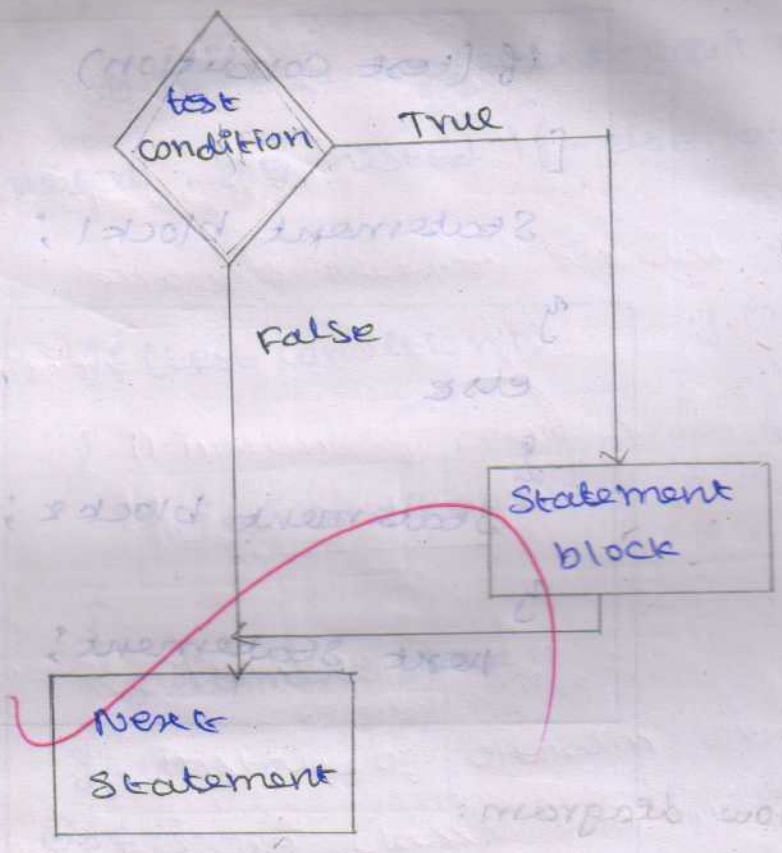
1. Simple if - statement
2. if - else statement
3. Nested if - else statement
4. else - if ladder statement
5. Switch statement.

1. Simple if - statement:

Syntax:

```
if (test condition)
{
  statement block;
}
next statement;
```

இதில் test condition-ஐ check பண்ண condition true-ஆக இருந்தால் statement block execute பண்ணும். பிறகு next statement execute ஆகும். condition false-ஆக இருந்தால் next statement execute ஆகும்.



eg:

```
m=50;
if(m>=40)
{
System.out.println("PASS");
}
```

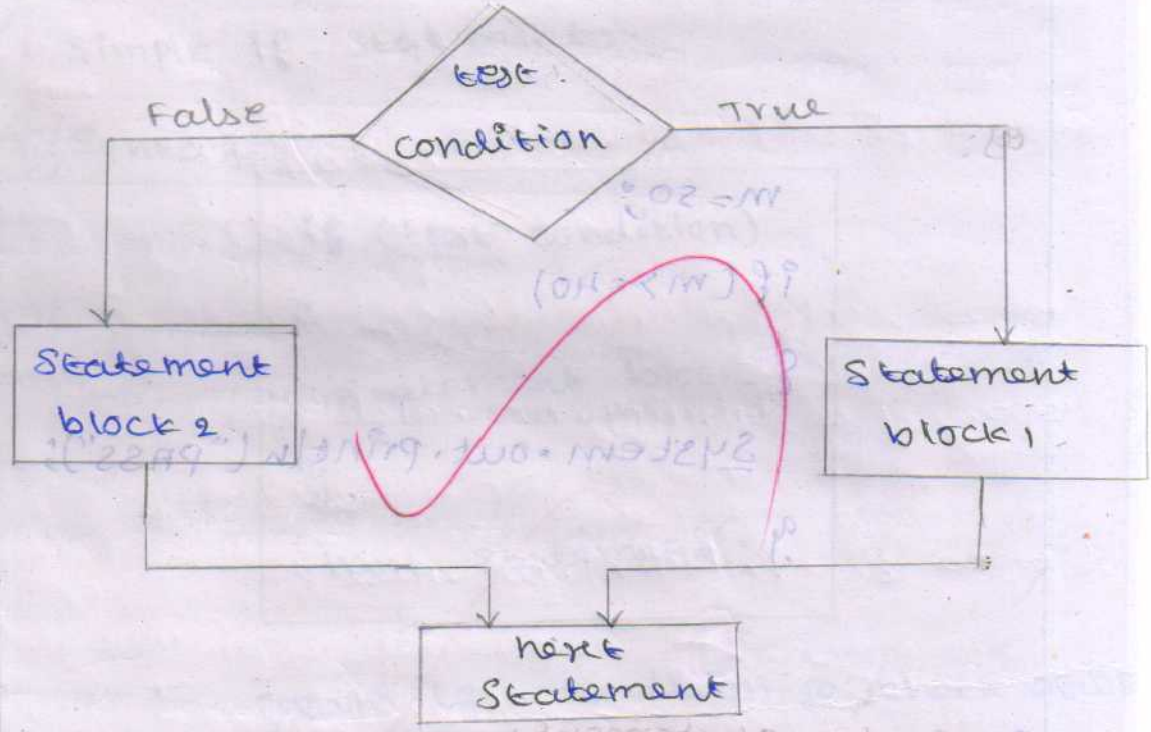
2. if-else statement.

இதில் test condition-ஐ check செய்கிறது condition true-ஆக இருந்தால் statement block1 ஐ execute செய்கிறது. false-ஆக இருந்தால் Statement block2 execute ஆகும்.

Syntax:

```
if (test condition)
{
  statement block1;
}
else
{
  statement block2;
}
  next statement;
```

flow deagram:



eg:

```
m = 30;
if (m >= 40)
{
System.out.println("PASS");
else
System.out.println("FAIL");
}
```

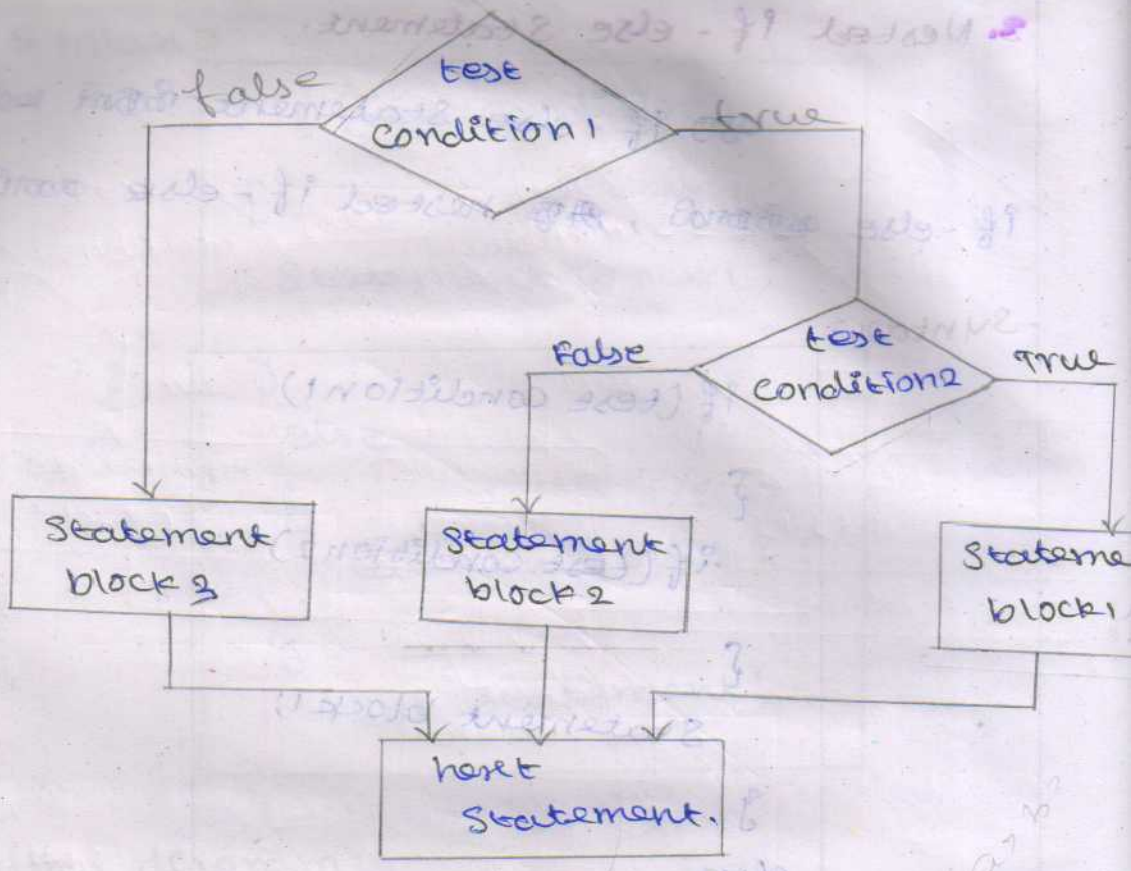
3. Nested if - else Statement.

ஒரு if - else statement-இன் மற்றொரு if - else வந்தால், அது nested if - else எனப்படும்.

Syntax:

```
if (test condition1)
{
  if (test condition2)
  {
    statement block1;
  }
else
  {
    statement block2;
  }
}
else
  {
    statement block3;
  }
  next statement;
```

இதில் test condition1 ஐ check செய்கிறது. அது false-ஆக இருந்தால் statement block 3 execute செய்கிறது. condition1 true ஆக இருந்தால் test condition2 ஐ true-ஆக இருந்தால் Statement block1 execute ஆகும். false-ஆக இருந்தால் Statement block 2 execute ஆகும்.



eg:

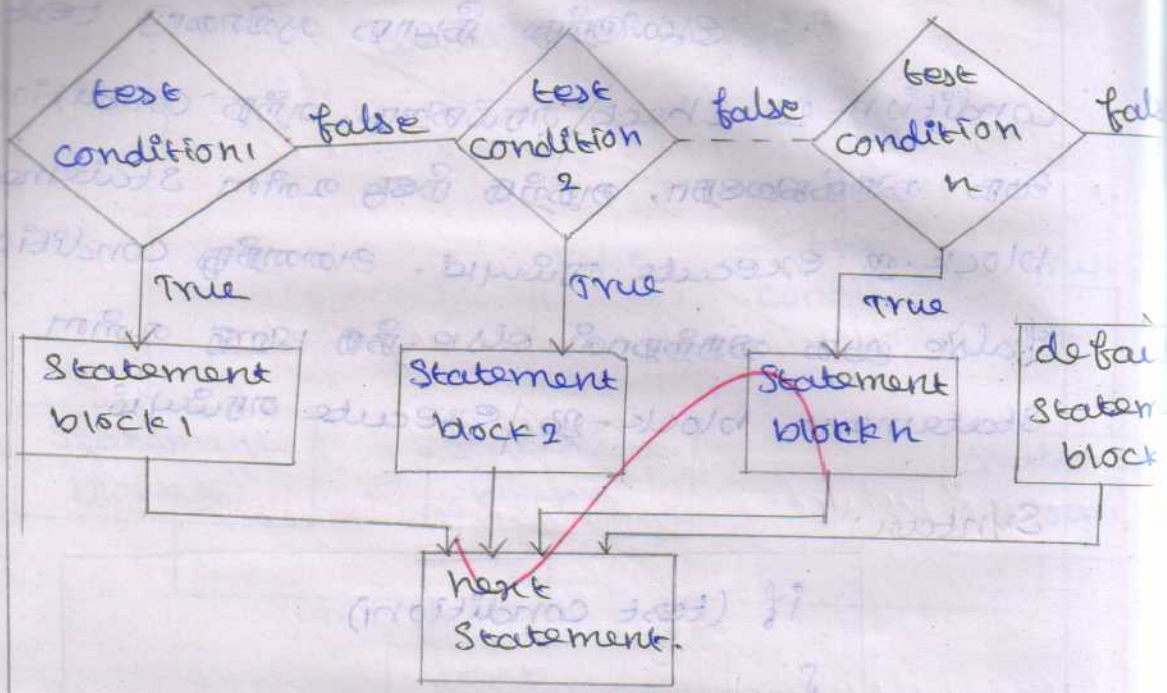
```
if(a>b)
{
  if (a>c)
  System.out.println("a is larger");
  else
  System.out.println("c is larger");
}
else
{
  if(b>c)
  System.out.println("b is larger");
else
  System.out.println("c is larger")
}
```

4. else if - ladder.

By (ஒவ்வொரு நிபந்தனை ஏனைய test condition-ஐ check செய்கிறது. எந்த condition true என கண்டறிகிறதோ, அதற்கு கீழே உள்ள statement block-ஐ execute செய்யும். அனைத்து condition-களும் false என கண்டால் else-க்கு பிறகு உள்ள statement block-ஐ execute செய்யும்.

Syntax:

```
if (test condition1)
{
  Statement block1
}
else if (test condition2)
 {
  Statement block2
 }
 ...
 ...
else if (test condition n)
  {
    Statement block n
  }
else
    default statement block
next statement.
```



eg:

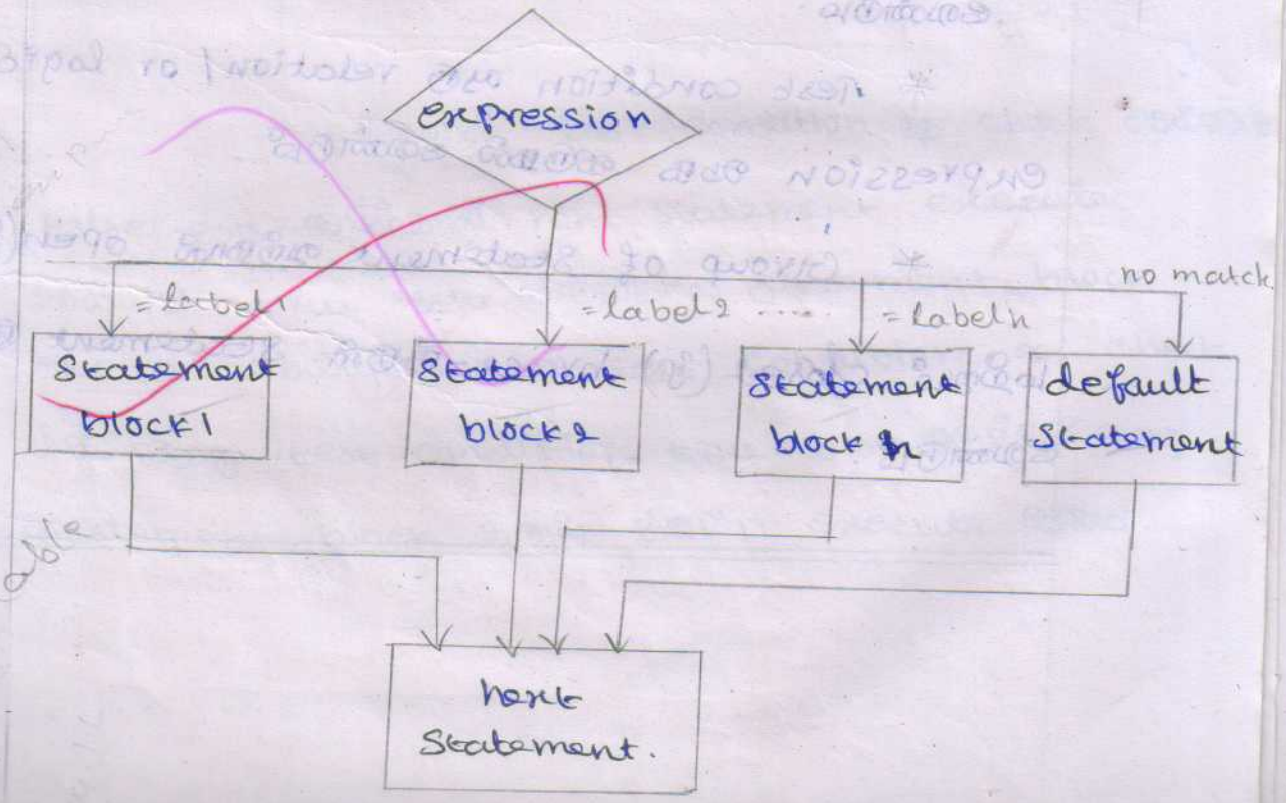
```
if (a>b)
System.out.println("a is larger");
else if (a<b)
System.out.println("b is larger");
else
System.out.println("a & b are equ
```

5. Switch statement:

ஒரு expression value-வை கண்டுபிடிக்கிறது. அந்த value-வை case-ல் உள்ள label உடன் compare செய்கிறது. எந்த label ஆகுதோ அதற்குரிய statement block-ஐ execute செய்யும். எந்த label-ம் match ஆகவில்லை என்றால் default என்ற உள்ள statement execute ஆகும்

Syntax:

```
switch (expression)
{
case label 1 : statement block1;
                break;
case label2 : statement block2;
                break;
- - - - - - - - -
. . - - - - . . .
case labeln : statement blockn;
                break;
default : statement block;
                break;
}
next statement;
```



Eg:

```
switch (n)
{
case 1: System.out.println ("ONE");
        break;
case 2: System.out.println ("TWO");
        break;
case 3: System.out.println ("THREE");
        break;
default : System.out.println("Invalid
                                data");
        break;
}
```

Rules:

* Test condition அடைப்பு குறியீடுக்குள் அமைய வேண்டும்.

* Test condition ஆக relation / or logical expression ஆக இருக்க வேண்டும்.

* Group of statement ஆரம்பம் open({) மற்றும் closed (}) brace-க்குள் statement இருக்க வேண்டும்.

3. Explain about looping statement.

Group of statement-ஐ condition-ஐ வைத்து repeat செய்வதற்கு looping statement பயன்படுகிறது.

3 types of looping statements.

1. while loop
2. do-while loop
3. for loop.

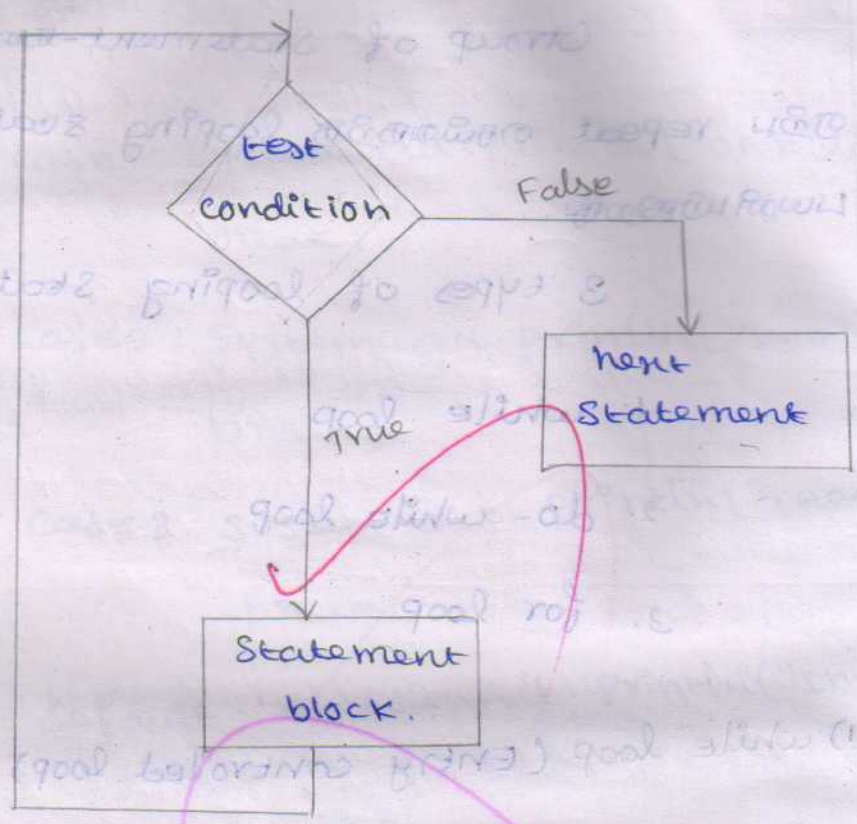
1) while loop (Entry controlled loop)

Syntax:

```
while (test condition)
{
  statement block;
}
  next statement;
```

இதில் test condition-ஐ check செய்கிறது false-ஆக இருந்தால் next statement execute செய்யும். true-ஆக இருந்தால் statement block execute செய்யும். பிறகு test condition-ஐ check செய்கிறது. test condition true-ஆக இருக்கும் வரை statement block திரும்ப திரும்ப execute ஆகும்.

Flow diagram:



eg:-

```
i=1;
while (i<=5)
{
System.out.println(i);
i++;
}
```

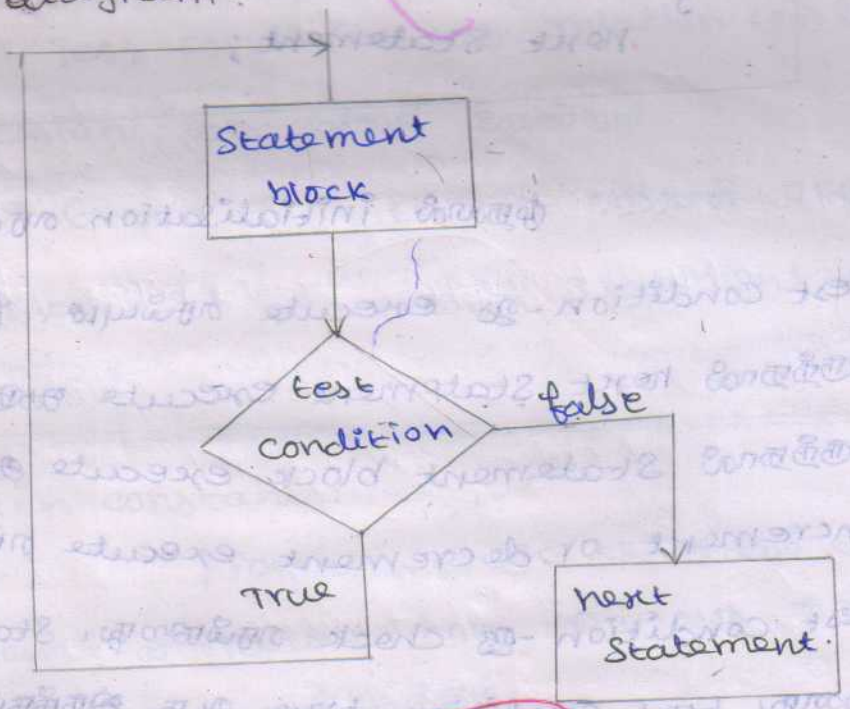
2) do-while loop (exit controlled loop)

Syntax:

```
do
{
  Statement block;
}
  while (test condition);
  next statement;
```

முதலில் Statement block-ஐ execute செய்கிறது. பிறகு test condition-ஐ check செய்யும். false-ஆக இருந்தால் next statement execute ஆகும். true-ஆக இருந்தால் மீண்டும் statement block execute ஆகும். test condition true ஆக இருக்கும் வரை statement block திரும்ப திரும்ப execute ஆகும்.

flow diagram:



eg:

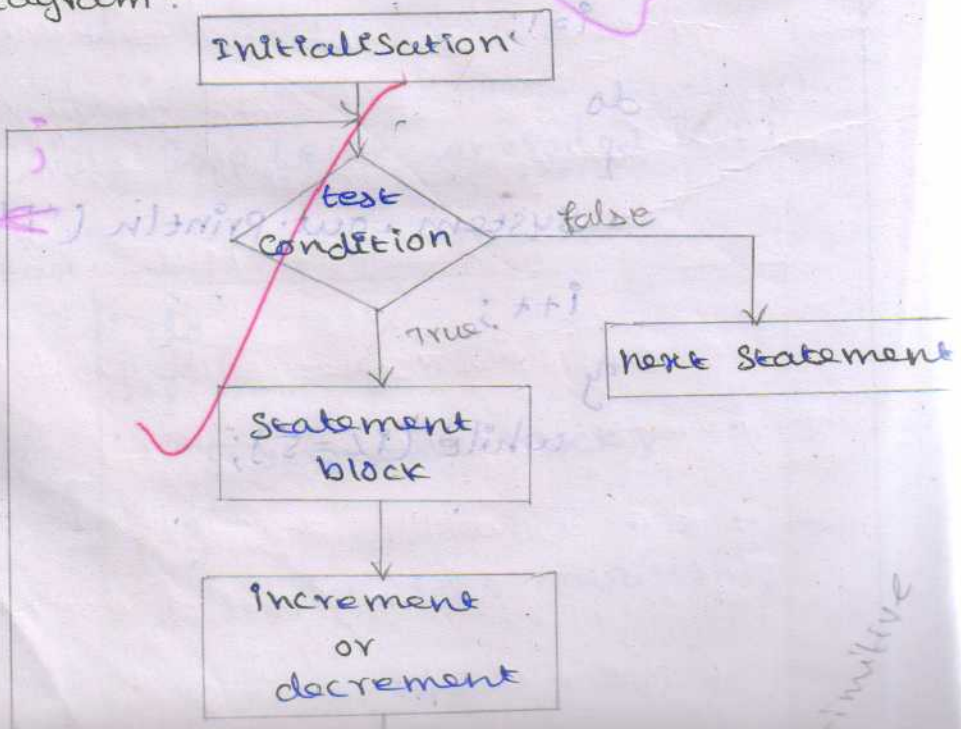
```
i=1;
do
{
  System.out.println (i);
  i++;
}
  while (i<=5);
```

3) for - loop

```
for (initialisation ; test condition ; increment
                                        decrement)
{
    Statement block ;
}
next statement ;
```

முதலில் initialisation ஏற்படும். பிறகு test condition - ஐ execute ஆகும். false - ஆக இருந்தால் next statement execute ஆகும். true - ஆக இருந்தால் Statement block execute ஆகும். பிறகு increment or decrement execute ஆன பிறகு test condition - ஐ check ஆகிறது. Statement block test condition true ஆக இருக்கும் வரை திரும்ப execute ஆகும்.

Flow diagram :



eg:

```
for (i=1; i<=5 ; i++)
{
System.out.println(i);
}
```

Rules:

* Test condition அடைப்பு கொடுக்கும் அவசியம் வேண்டும்.

* Test condition ஒரு relation (அ) logical expression -ஆக இருக்க வேண்டும்.

* Group of statement வந்தால் open ({) மற்றும் ~~for~~ closed (}) brace-க்குள் statement கொடுக்க வேண்டும்.

Subscript
sw

II unit

Define Array. Explain how will you create one. dimensional array.

Array என்பது ஒன்றுக்கொன்று தொடர்புடைய data தொகுப்பை ஒரே மாதிரி பயன்படுத்தி பதிய செய்யும் தொகையாகும்.

One dimensional array

Arrayன் ஒரே subscript மட்டும் அது one dimensional array எனப்படும்.

Creating an Array

① Array ஐ declare செய்தல்

```
datatype    arrayname[];
```

datatype - valid Java datatype

arrayname - name of the array

Eg: `int x[];`

② Memory space allocate செய்தல்

```
arrayname = new datatype[size];
```

new - memory allocate செய்வதற்கான operator

Eg: `x = new int[10];`

③ Storing values in the array

```
arrayname [subscript] = value;
```

eg. x[1] = 40;

Array ல declare பண்ணும் போது value assign பண்ணலாம்.

```
datatype arrayname [ ] = { value1, value2 .... valuen}
```

eg int y[ ] = { 65, 70, 80};

② Explain about Vector.

Vector என்பது ஒரு class. இது dynamic array create பண்ண பயன் படுகிறது. vector ல different types of objects store பண்ணலாம். Vector class ஆனது java.util என்ற package ல் உள்ளது.

Creation

1) creating without size

```
Vector vectorname = new Vector();
```

இந்த method ஆனது empty Vector create பண்ணுகிறது. Vector ன் size 10 ஆகும்.

eg Vector V1 = new Vector();

2) creating with size

Vector vectorname = new Vector(n);

eg Vector V2 = new Vector(7);

(3) creating with size and increment.

Vector vectorname = new Vector(n, increment);

இந்த method n size-ன் vector-ஐ create செய்கிறது. size மாறும்போது increment அளவுக்கு change ஆகிறது.

eg Vector V3 = new Vector(7, 2);

Methods

Method	Use	Example.
1) capacity()	இந்த method vector-ன் எத்தனை objects-ஐ store செய்யலாம் என்பதை கண்டறிய பயன்படுகிறது	V1.capacity()
2) size()	இந்த method vector-ல் store செய்துள்ள objects எண்ணிக்கையை தருகிறது.	V1.size()
3) addElement(object)	இந்த method vector-	

	கூறுகளை Objectஐ add செய்கிறது.	
4. insertElement At(object, n)	இந்த method Vectorன் nth positionஇல் Objectஐ செருகிறது	V1.insertE At("CE",
5. removeElement(object)	இந்த method குறிப்பிட்ட Objectஐ vectorல் இருந்து remove செய்கிறது.	V1.removeEle ent("CE")
6. removeElement At(n)	இந்த method nth positionஇல் உள்ள Objectஐ remove செய்கிறது.	V1.removeElement At(4);
removeAllEle ments()	இந்த method Vectorன் உள்ள அனைத்து elementsஐயும் remove செய்கிறது	V1.removeAllEle ments();
elementAt(n)	இந்த method nth positionஇல் உள்ள elementஐ return செய்கிறது	V1.elementAt(2);
copyIntoArray name)	இந்த method Vectorல் உள்ள elementsஐ arrayல் copy	V1.copyInto(a);

Arraylist என்பது ஒரு class. இது வெண்டுமென்றே dynamic array-யை உருவாக்கவும். different types of objects-ஐ இதில் store செய்யவும். இந்த class java.util என்ற package-ல் உள்ளது.

Creation

1) Creating without size

```
ArrayList name = new ArrayList();
```

இந்த method ஒரு empty arraylist-ஐ create செய்கிறது.

eg ArrayList a1 = new ArrayList();

2) Creating with size

```
ArrayList name = new ArrayList(n);
```

இந்த method n size-ல் ஒரு arraylist-ஐ create செய்கிறது.

eg ArrayList a2 = new ArrayList(4);

Methods

method	use	Example
1) size()	இந்த method array list-ல் store செய்துள்ள objects-ன் எண்ணிக்கையை தருகிறது	a1.size();

2) add (object)	இந்த method arraylistன கடைசியில் objectஐ சேர்க்கிறது.	
3) add(n, object)	இந்த method objectஐ nth positionல் சேர்க்கிறது	al.add(2,
4) remove(n)	இந்த method nth positionல் உள்ள objectஐ remove செய்கிறது.	al.remove(2)
5) clear()	இந்த method array listன் காணப்படும் அனைத்து elementகளையும் remove செய்கிறது	al.clear();
6) get(n)	இந்த method array listன் nth positionல் உள்ள elementஐ return செய்கிறது.	al.get(3);
7) set(n, object)	இந்த method nth positionல் உள்ள objectஐ கொடுக்கப்பட்ட objectஆல் replace செய்கிறது	al.set(2, "ECE");

at is wrapper class? Explain

class ஆனது Primitive datatypeன

object [illegible]
typeக்கு convert செய்ய பயன்படுகிறது. இந்த
class அனைத்தும் java.lang என்ற packageல் உள்ளது. (4)

primitive data type	Classname
int	Integer
float	Float
byte	Byte
short	Short
char	Char
double	Double
boolean	Boolean.

wrapper classகளை பயன்படுத்தி கீழ் உள்ள conversionகளை செய்யலாம்.

1) converting primitive datatype to object
2) converting object to primitive datatype
3) converting primitive number to string object
4) converting string object to primitive number
5) converting string object to number object.

1) <u>converting primitive datatype to object</u>

primitive datatypeஐ objectஆக மாற்றுவதற்கு
[illegible] பயன்படுகிறது.

Classname objectname = new classname(value);

eg: convert int to Integer object

Integer K1 = new Integer(60);

2) converting object to primitive datatype.

object ను primitive datatype లోకి మార్చడానికి typeValue() ఉపయోగిస్తాము.

datatype variablename = objectname.typeValue();

eg: convert float object to float datatype.

float s1 = f1.floatValue();

3) converting primitive number to string object

primitive datatype ను string object లోకి మార్చడానికి toString() ఉపయోగిస్తాము.

String objectname = classname.toString(value);

eg: convert int to string object

String s2 = Integer.toString(67);

4) converting String object to primitive number

String object ను primitive number లోకి మార్చడానికి parseType() ఉపయోగిస్తాము.

datatype variable = classname.parsename(string object);

5) Converting String object to number object

String object ஐ number object ஆக மாற்றுவதற்கு value Of() பயன்படுகிறது.

```
Classname objectname = classname.valueOf(String object);
```

eg. convert string object to float object

```
Float f2 = Float.valueOf(s1);
```

3 marks

1) Give the difference between Array and ArrayList

Array	ArrayList
1) இதன் size மாறாது.	இதன் size மாறக்கூடியது.
2) இதில் ஒரே மாதிரியான data வை மட்டும் store செய்யலாம்.	இதில் எல்லா மாதிரியான objectஐயும் store செய்யலாம்.
3) இதில் elementஐ insert மற்றும் delete செய்ய முடியாது.	இதில் elementஐ insert மற்றும் delete செய்ய முடியும்.

III unit

(1) Explain about String.

String என்பது double quotations-ல் உள்ள group of characters ஆகும்.
Java-ல் string என்பது String class-ல் object ஆக treat செய்யப்படுகிறது. String class Java.lang என்ற package-ல் உள்ளது. String object-ல் content-ஐ மாற்ற முடியாது.

creation

	Syntax	Example
1. creating a empty string	String name = new String();	String s1 = new String();
2. creating string with characters	String name = new String (value);	String s2 = new String ("good");
3. creating string with another string	String name = new String (string object);	String s3 = new String (s2);
4. creating a string using substring	String name = new String (String, m, n); m - Starting position n - number of characters	char x[] = {'p', 'r', 'o', 'g'}; String s4 = new String (x, 2, 3); (s4 = rog)

Methods

`stringobject.methodname();`

`String S1 = new String("Example");`

Method	use	example
1. length()	string-ன் length-ஐ கண்டுபிடிக்க பயன்படுகிறது.	S1.length()
2. toLowerCase()	string-ன் உள்ள characters-ஐ lower case ஆக மாற்றுகிறது.	S1.toLowerCase()
3. toUpperCase()	string-ன் உள்ள characters-ஐ upper case ஆக மாற்றுகிறது.	S1.toUpperCase()
4. trim()	string-ன் முன்பு மற்றும் பின்பு உள்ள blank spaces delete செய்கிறது	S1.trim()
5. concat(S1)	S1 ஐ calling string உடன் சேர்க்கிறது.	S2.concat(S1)
6. equals(S1)	S1 மற்றும் calling string equal ஆக உள்ளதா என்று check செய்கிறது. இரண்டும் equal ஆக இருந்தால் true return செய்யும். இல்லை என்றால் false return செய்யும்	S2.equals(S1)
7. substring(n)	nth position-லிருந்து string-ஐ return செய்கிறது.	S1.substring(5)
8. indexOf(ch)	string-ன் char first position-ஐ return செய்கிறது	S1.indexOf('a')

2. Explain about stringbuffer class. (2)

stringbuffer class variable length strings create பண்ண பயன்படுகிறது. இந்த class java.lang என்ற package உள்ளது. stringbuffer object ல் contents மாற்றலாம்.

creation

	Syntax	Example
1. creating a empty string buffer (16 memory space allocate ஆகும்)	StringBuffer name = new StringBuffer();	StringBuffer n1 = new StringBuffer();
2. creating a string buffer of size n	StringBuffer name = new StringBuffer(n);	StringBuffer n2 = new StringBuffer (20);
3. creating a stringbuffer with initial value	StringBuffer name = new StringBuffer(string);	StringBuffer n3 = new StringBuffer("Book");

Methods

stringbufferobject.methodname();

StringBuffer b1 = new StringBuffer("college");

Method	use	Example
1) length()	String Buffer ல் actual length கண்டுபிடிக்க பயன்படுகிறது.	b1.length();
2) capacity()	StringBuffer ல் (maximum length) capacity கண்டுபிடிக்க பயன்படுகிறது	b1.capacity();

3) setLength(n)	string buffer object-ன் length-ஐ set செய்ய பயன்படுகிறது	b1. setLength(3);
4) append(s1)	s1 ஐ string buffer object-ன் கடைசியில் சேர்க்கிறது	b1. append(s1);
5) reverse( )	Stringbuffer object-ஐ reverse செய்ய பயன்படுகிறது	b1. reverse( )
6) insert(n, s1)	stringbuffer object-ன் nth position-ல் s1-ஐ சேர்க்கிறது	b1. insert(0, "adv");
7) setCharAt(n, ch)	stringbuffer object-ன் nth position-ல் உள்ள character-ஐ ch ஆக string செய்கிறது.	b1. setCharAt(5, 'E');

3. Explain about Class and object.

class என்பது user defined data type. இதில் data-யும் அதற்கு சம்பந்தமான methods-ஐயும் உள்ளடக்கி

```
class classname
{
  datatype variable1;
   - - - - -
   - - - - -
  datatype variablen;
  datatype methodname(parameters)
  {
    Statements
  }
- - -
}
```

1) class-ல் declare செய்யும் variables-ம் instance variable எனப்படும்.

2) class-ல் declare செய்யும் methods-ம் instance method எனப்படும்.

3) class-ல் உள்ள variable மற்றும் methods-ம் members எனப்படும்.

Eg:

```
class Point
{
   int x, y;
  void read (int a, int b)
   {
     x = a;
     y = b;
   }
  void display()
   {
     System.out.println("x =" + x);
     System.out.println("y =" + y);
   }
}
```

Object

class type variables-ம் object எனப்படும். object create செய்யும்போது instance variables-க்கு memory space allocate செய்யப்படும்.

creating object

1) Declare object

```
Classname object1, object2 ..... objectn;
```

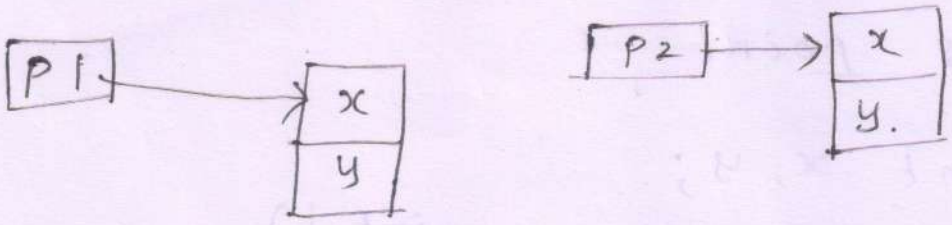
Eg: Point P1, P2;

2) Allocate memory space

new operator பயன்படுத்தி objects-களுக்கு memory space allocate செய்யப்படுகிறது

```
objectl = new classname();
object2 = new classname();
- - - - -
- - - - -
objectn = new classname();
```

eg. p1 = new point();
p2 = new point();



Accessing class member.

object ஒவ்வொரு class members access செய்யலாம்.

```
object . variable
object . methodname();
```

eg. p1.x = 10; p1.display();

4. Explain about constructor

Constructor என்பது class உள்ள special method. இது instance variable க்கு தேவையான மதிப்பை கொடுப்பதற்கு பயன்படுகிறது. constructor ஆனது object உருவாக்கும் போது தானாகவே call. ஆகும்.

```
constructorname (arguments)
{
    Statements
}
```

eg

```
point ()
{
  x = 40;
  y = 60;
}
```

Rules

1. constructorname classname என்ற பெயர்

2. constructor-க்கு return type இருக்காது.

இரண்டு வகையான constructor உள்ளன.

(1) Constructor without arguments or Default constructor

Constructor-ல் argument இல்லாமல் எழுதுவது இந்த default constructor எனப்படும்.

(2) constructor with arguments or Parameterised Constructor

Constructor-ல் argument இருந்தால், அது Constructor with argument எனப்படும்.

eg (1)
```
point ()
{
    x = 40;
    y = 50;
}
```

(2)
```
point (int a, int b)
{
    x = a;
    y = b;
}
```

5. Explain 1) constructor overloading 2) Nesting of methods 3) this keyword.

Constructor overloading

ஒரே class-ல் ஒன்றுக்கு மேற்பட்ட Constructor-கள் இருந்தால் அது constructor overloading எனப்படும்.

eg:
```
class point
{ int x, y;
  point ()
  { x = 100;
    y = 40;
  }
  point (int a, int b)
  { x = a;
    y = b;
```

eg2:
```
class Sum
{ int x, y;
  sum ()
  { x = 10;
    y = 20;
  }
  int total ()
  { return (x+y);
  }
  void display ()
  { S.o.p("sum=" +
      total());
  }
}
```

② Nesting of methods

class ன் ஒரு method, அது class ன் ஒரு மற்ற method ஐ call செய்வது அது nesting of methods எனப்படும்.

இதற்கு கீழே ஒரு example ன், display() method ஆனது total() method ஐ call செய்கிறது.

③ this keyword : இது ஒரு implicit pointer ஆகும். this keyword ஆனது hidden class instance variables ஐ access செய்யவும். இது pointer methods ஐ call செய்யும் objects ன் addresses ஐ சேமிக்கும்.

eg.

```
class Example
{
    int x = 10;
    void display()
    {
        int x = 40;
        S.o.p("x =" + x);
        S.o.p("x =" + this.x);
    }
}
```

6. Explain about static members.

class ன் ஒரு variable அல்லது method ஐ static என்ற keyword ஆனது, அது static member ஆக்கும். syntax.

Characteristics

1) static methods ~~object~~ class ன் பெயரிலிருந்து call செய்ய வேண்டும்.

```
Classname.staticmethodname();
```

2) static method static variableஐ மட்டும் access செய்ய முடியும்.

3) static variable களுக்கு common memory allocate செய்யப்படும்.

4) static member க்கு நேரடியாக this அல்லது super போன்ற keyword பயன்படாது.

Syntax

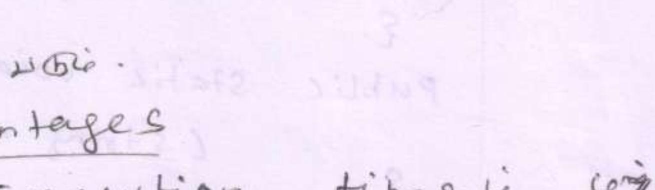
```
class name
{
static datatype variable;
...
static datatype method( )
{ ...
}
}
```

eg

```
class Example
{
  static int x = 40;
  static void print( )
  {
    s.o.p ("x =" + x);
  }
}
```

4. Explain about inheritance. (or) Explain the different types of inheritance.

Inheritance என்பது இருக்கின்ற classல் இருந்து புதிய class கள் உருவாக்கும் ஒரு முறை. இருக்கின்ற class super (அ) Base (அ) parent class என்றும் புதிய class sub (அ) derived (அ) child class என்றும் அழைக்கப்படும்.



Advantages

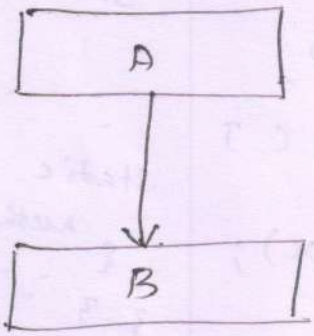
1) Execution time ம் memory space குறைகிறது.
2) Programன் நம்பகத்தன்மையை அதிகரிக்கிறது.
3) Program உருவாக்குவதற்கான நேரம் குறைகிறது.

Types of inheritance

① Single inheritance ② Hierarchical inheritance
③ Multilevel inheritance ④ Multiple inheritance

Single inheritance

One super class and one sub class is single inheritance.



Syntax

```
class name2 extends name1
{
  - - - - -
  - - - - -
}
```

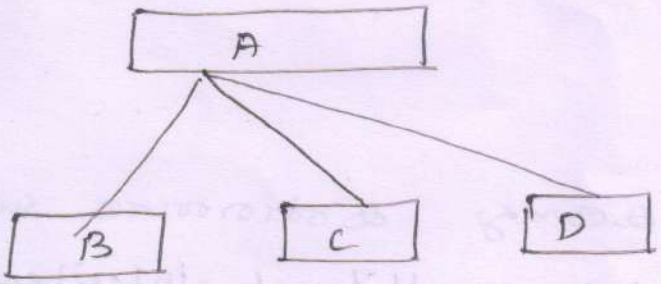
name2 – subclass name
name1 – superclass name
class, extends – keywords.

eg.

```
class First
{
  int x;
  First(int a)
  { x = a;
  }
  void print()
  { s.o.p(x);
  }
}
class Second extends First
{
  int y;
  Second(int p, int q)
  {
    super(p);
    y = q;
  }
  void print1()
  {
    s.o.p(y);
  }
}
class Sinheritance
{
public static void main(String args[])
{
  Second s1 = new Second(5, 20);
  s1.print();
  s1.print1();
}
}
```

② Hierarchical inheritance

ஒரே super class-ல் இருந்து, ஒன்றுக்கும் மேற்பட்ட sub class-கள் inherit செய்யப்பட்டால், அது Hierarchical inheritance எனப்படும்.



class, extends - keywords

name1 - superclass-ன் பெயர்

name2, name3 .... namen - sub classகளின் பெயர்கள்

Syntax

```
class name2 extends name1
{
  _ _ _
  _ _ _
}
class name3 extends name1
{ _ _ _
  _ _ _
}
- - - - -
- - - - -
class namen extends name1
{
  _ _ _
  _ _ _
}
```

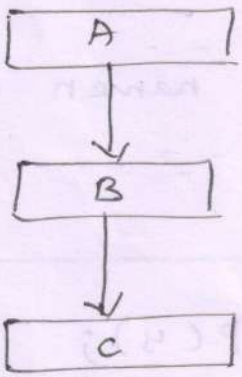
ex

```
class First
{ int x;
  First(int a)
  { x = a;
  }
  void print()
  {
    S.o.P(x);
  }
}
class Second extends First
{ int y;
  Second(int p, int q)
  {
    super(p);
    y = q;
  }
  void Print1()
  {
    S.o.P(y);
  }
}
class Third extends First
{ int k;
  Third(int p, int q)
  { super(p);
    k = q;
  }
  void Print2()
  { S.o.P(k);
  }
}
class Hinheritance
{
  P s v m( )
{
Second S1 = new Second(
```

```
Third t1 = new Third(40, 60);
s1.print1();
t1.print2();
s1.print();
}
}
```

③ Multilevel inheritance

ஒரு sub class ஐ கொண்டு கொண்டாக sub class உருவாக்கப்பட்டால், அது multilevel inheritance எனப்படும்.



class, extends - keywords

name1 - superclass ன் பெயர்

name2, name3 ... namen - sub class களின் பெயர்கள்

Syntax

```
Class name2 extends name1
{ - - -
  - - -
}
Class name3 extends name2
{ - - - -
  - - -
}
- - -
- - -
Class namen extends namen-1
{ - - - -
  - - -
}
```

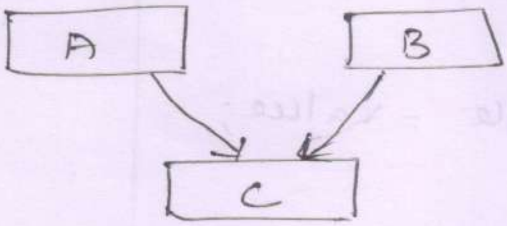
eg.

```
class First
{ - - -
  - - -
}
class Second extends First
{ - - -
  - - -
}
class Third extends Second
{ - - -
  - - -
```

```
class Minheritance
{
  p s v m()
{
  Third t1 =
    new Third(10, 40,
              60);
  t1.print2();
}
}
```

(4) Multiple inheritance

ஒன்றிற்கு மேற்பட்ட super classகளில் இருந்து ஒரே ஒரு sub class உருவாக்கப்பட்டால், அது multiple inheritance எனப்படும். இதை implement செய்வதற்கு interface பயன்படுகிறது.

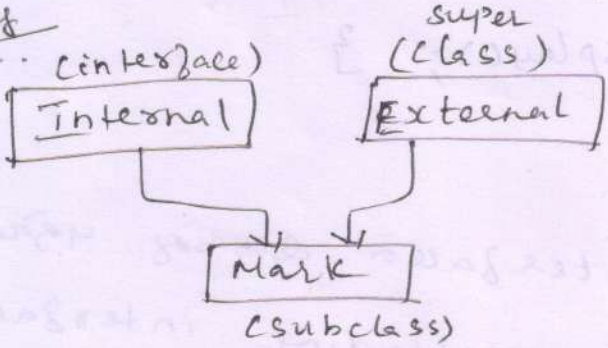


extends, Class, implements – keywords
name1 – super classன் பெயர்
name2 – subclassன் பெயர்
interface1, ... interfacen – interfaceகளின் பெயர்கள்

syntax

```
class name2 extends name1 implements interface1,
                        interface2 .....interfacen
{
  - - - -
  - - - -
}
```

eg




```
interface Internal
{
  -- --
  -- --
}
class External
{ __ __
  __ __
}
```

```
class Mark extends External
      implements Internal
{
  - - - -
  - - - -
}

class Minheritance
{
  p s v m ( )
  {
  Mark m1 = new Mark
               (...);
  - - - -
  }
}
```

8. Define interface. Explain how it is created, extended and implemented with example.

Interface என்பது user defined datatype. இதில் abstract methods மற்றும், final variables மட்டும் காணப்படும்

syntax

```
interface name
{
  final datatype variable = value;
   - - -
  - - -
  abstract datatype methodname();
   - - -
  - - -
}
```

eg:

```
interface Area
{
  float Pi = 3.14f;
  abstract void display();
}
```

eg2

```
interface Y extends Area
{
  - - - -
  - - -
}
```

extending interface

ஒரு interface ஆனது மற்றொரு interface உடன் extending interface என்ற keyword பயன்படுத்தி extend செய்யப்படுகிறது.

```
interface name2 extends name1, ... namen
  {
    - - -
    - - -
  }
```

eg2

implementing interface

Define செய்யப்பட்ட interface ஆனது class ல் implementing keyword பயன்படுத்தி implement செய்யப்படுகிறது.

interface ல் உள்ள variables மற்றும் methods ஐ class ல் inherit செய்ய வேண்டும். interface ல் உள்ள methods ஐ class ல் கண்டிப்பாக define செய்ய வேண்டும்.

```
class name1 implements interface1, interface2 ---
  {
    ----
    ---
  }
```

---

eg:

```
interface Area
{
  float pi = 3.14f;
  abstract void displays();
}
class Circle implements Area
{
  int r;
  circle (int a)
  {
    r = a;
  }
Public void displays()
  {
    S.o.p("Area =" + (Pi * r * r));
  }
}
```

(9) Explain about final variable, final method and final class.

final variable : variable declaration ல் முன்பாக final என்ற keyword வந்தால், அந்த variable final variable என்றழைக்கப்படும். final variable ன் value வை மாற்ற முடியாது

```
final datatype variable = value;
```

eg: final double Pi = 3.14;

final method: method definitionக்கு முன்பாக final என்ற keyword வருகின்றன, அது final method எனப்படும். final methodஐ subclassல் override செய்ய முடியாது.

```
final datatype methodname()
    {
      ---
      ---
    }
```

eg:
```
final void display()
    {
      S.o.P(x);
    }
```

final class: class declarationக்கு முன்பாக final என்ற keyword வருகின்றன, அது final class எனப்படும். final classன் கீழுள்ள subclass உருவாக்க முடியாது.

```
final class name
  {
    ---
    ---
  }
```

eg:
```
final class point
  {
    int x, y;
    ---
    ---
  }
```

(10) Explain about abstract method & abstract class.

abstract method: method declarationக்கு முன்பாக abstract என்ற keyword வருகின்றன, அது abstract method ஆகும். abstract methodக்கு definition கிடையாது. இதை subclassல் கண்டிப்பாக override செய்ய வேண்டும்.

```
abstract  datatype  methodname();
```

eg. abstract void Print();

abstract class : class declaration-ல் முன்பாக abstract keyword பயன்படும், இது abstract class எனப்படும். abstract class-ன் object create செய்ய முடியாது.

```
abstract class name
{
  ---
  ---
}
```

eg.
```
abstract class X
{
  ---
  ---
}
```

(11) write short notes overriding methods.

super class-ன் define செய்த method-ஐ subclass-ன் redefine செய்வது overriding method எனப்படும். Super class-ன் உள்ள method-ஐ call செய்ய keyword பயன்படுத்தி வேண்டும் எனில் super எனும் வேண்டும்.

eg.
```
class First
{ int x;
  ---
  void display()
  { S.O.P(x);
  } }
class Second extends First
{ int y;
  ---
  void display()
  {
    super.display();
    S.O.P(y);
  }
}
class Sample
{
  P S Vm ( )
  {
    Second S1 =
      new Second();
    S1.display();
  }
}
```

(12) Explain about visibility control.

ஒரு class ன் உள்ள variable மற்றும் method களை மற்ற classகள் (அ) subclassகள் (அ) packageகள் பயன்படுத்த வேண்டுமென்றால், access modifier களைக் கொடுக்க வேண்டும். access modifierகள் ஐந்து வகைகள்

1) public 2) private 3) protected

4) friendly 5) private protected.

(modifier எதுவும் தரவில்லை என்றால், friendly என்று எடுத்துக் கொள்ளும்)

Syntax

```
accessmodifier datatype variable;
accessmodifier datatype methodname()
{
----
----
}
```

	private	public	protected	friendly	private protected.
same class	yes	yes	yes	yes	yes
sub class in the same package	NO	yes	yes	yes	yes.
other class in the same package	NO	yes	yes	yes	NO
Sub class in other package	NO	yes	yes	NO	yes
other class in other Package	NO	yes.	NO	NO	NO.

eg. private int xy;

protected void print()
{

}

IV unit

1.—Define Package. Explain how will you create and access a package.

Package என்பது classes மற்றும் interfaceகளின் தொகுப்பு ஆகும்.

Creating Package

1. Packageஐ declare செய்ய வேண்டும்.

```
Package  Packagename;
```

2. Packageன் classஐ Public ஆக define செய்ய வேண்டும்.

```
Public  class name
{
  .....
  .....
}
```

3. Package nameன் அதே Subdirectoryஐ create செய்ய வேண்டும்.

```
C:\jdk1.7\bin>md  Packagename↵
C:\jdk1.7\bin>cd  Packagename↵
C:\jdk1.7\bin\packagename>
```

4. Subdirectoryன் Programஐ Save செய்யவேண்டும்.

5. Programஐ compile செய்ய வேண்டும்.

Eg:

```
Package college;
Public class Student
{
   String name;
   int regno;
   Public Student (String n1, int r)
   {
      name = n1;
      regno = r;
   }
   Void display()
   {
      System.out.println ("Name =" + name);
      System.out.println ("Regno=" + r);
   }
}
```

இந்த Program college என்ற ஒரு subdirectory create செய்து அதில் store செய்ய வேண்டும்.

i-e c ---> md college

c ---> cd college

c ... \college >

Accessing a Package

Package இரண்டு வழிகளில் access செய்யலாம்.

1. By Specifying the fully qualified class name
2. By using import statement.

(1) By specifying the fully qualified classname.

```
package1.[package2].[package3].classname.obj;
```

eg

```
java.util.Vector.obj;
```

(2) By using import statement.

```
import package1.[package2].....classname;
        or
import package1.[package2].....*;
```

eg

```
import college.Student or import college.*;
```

eg

```
import college.*;
class Example
{
  public static void main(String args[])
  {
    Student s1 = new Student("Jeya", 4025);
    s1.display();
  }
}
```

(2) Define Applet. Explain how an applet is created and executed.

Applet என்பது internet application-க்கான window சார்ந்த Java program ஆகும்.

Creating an Applet.

Applet-ஐ create செய்வதற்கு Applet மற்றும் Graphics classes தேவை.

Applet class

இந்த class-ஐ extend செய்து புது applet உருவாக்கலாம். இந்த class Java.applet என்ற package-ல் உள்ளது. இதில் கீழே உள்ள methodsம் உள்ளன.

1) init() - applet-ன் variables-ஐ initialise செய்ய பயன்படுகிறது.
2) start() - applet-ஐ run செய்ய பயன்படுகிறது.
3) stop() - applet-ன் execution-ஐ stop செய்ய பயன்படுகிறது.
4) destroy() - applet-ஐ memory-லிருந்து நீக்க பயன்படுகிறது.
5) paint() - graphical output-ஐ applet-ல் display செய்ய பயன்படுகிறது.

Graphics class

இந்த class-ல் உள்ள drawString() method output-ஐ display செய்ய பயன்படுகிறது. இந்த class Java.awt என்ற package-ல் உள்ளது.

Syntax

```
import java.awt.*;
import Java.applet.*;
Public class classname extends Applet
{
  ....
  ....
   public void init()
   { ....
     ....
   }
  ....
   Public Void paint (Graphics g)
      {  ....
      }
}
```

eg:
```
import java.awt.*;
import java.applet.*;
public class Applet6 extends Applet
{
  public void paint(Graphics g)
  {
    g.drawString("welcome", 20, 80);
  }
}
```

Executing an applet

Applet ஐ கீழ்காண் படிகளின் execute செய்யலாம்.

1) using appletviewer command
2) using webbrowser command.

using appletviewer command

1) applet tag உடன் program ஐ சேர்க்கவேண்டும்.
2) பிறகு அந்த program உடன் .java என்ற extension உடன் Save செய்ய வேண்டும்.
3) Javac command மூலம் compile செய்யவேண்டும்.
4) appletviewer command உடன் பயன்படுத்தி run செய்ய வேண்டும்.

```
appletviewer classname.java
```

eg:
```
// <applet code="Applet6" width=400 height=600>
// </applet>
import java.awt.*;
```

```
import java.applet.*;
public class Applet6 extends Applet
{
   public void paint(Graphics g)
   {
      g.drawString("GOOD", 30, 80);
   }
}
```

2) using webbrowser

1) Applet programs .java extension save எனது, compile எனவே வேண்டும்.

2) HTML program applet tag பயன்படுத்தி வேண்டும். இந்த HTML programs bin directory .html extension save எனவே வேண்டும்.

3) webbrowser html files run எனவே வேண்டும்.

ex.
```
<html>
<body>
<applet code = "Applet6" width = 400
                         height = 400>
</applet>
</body>
</html>
```

(3) Explain about Applet Life cycle.

Applet life cycle ல் ஐந்து states உள்ளன.

1) Initialization state
2) Running state
3) Idle or stopped state
4) Dead state
5) Display state.

1) Initialization state.

இது applet life cycle-ன் முதல் state ஆகும். இது variables-ஐ initialise என்ற பயன் படுத்தி. init() என்ற method-ஐ பயன்படுத்தி initialise என்பார்கள்.

```
public void init()
{
    . . . .
    . . . .
    . . . .
}
```

Rules

i) இந்த state applet life cycle-ன் முதல் முறை மட்டுமே நடக்கும்.

ii) இந்த method-ஐ applet program-ல் override என்ற வேண்டும்) @

2) Running state

இது applet life cycle-ன் இரண்டாவது state ஆகும். இது initialisation state-ன் முடிவாகவே நடக்கும். start() method (or) back arrow பயன்படுத்தும் போது இந்த state தொடங்கும்.

```
public void start()
{
    . . . .
}
```

Rules

i) இந்த state applet life cycle-ல் எத்தனை முறை

வேண்டுமானாலும் பயன்படுத்தலாம்). (b)

ii) (a)

3) Idle or stopped state.

இந்த state applet life cycleன் மூன்றாவது state ஆகும். இது stop() method ஐ தொடர்ந்து arrow பயன்படுத்தும்போது இந்த stateக்கு வரும்.

```
public void stop()
{
. . .
. . .
}
```

Rules

i) இந்த method running stateன் மேற்கே பயன்படுத்த வேண்டும்.

ii) (a)

iii) (b)

4) Dead state.

இந்த state applet life cycleன் நான்காவது state ஆகும். destroy() method பயன்படுத்தும்போது இந்த stateக்கு வரும். இந்த stateன் applet அனைத்து memoryயிலிருந்து நீக்கிவிடும்.

```
public void destroy()
{
. . . .
. . . .
}
```

Rules

i) இந்த state எதுவும் மேற்கே இருக்கும்.

ii) (a).

5) Display state.

இந்த state applet life cycleன் ஐந்தாவது state ஆகும். Appletன் ஏதாவது display தையும் போது இந்த stateக்கு வரும். paint() method பயன்படுத்தும்போது

```
public void paint(Graphics g)
{
. . . .
. . . .
}
```

Rules

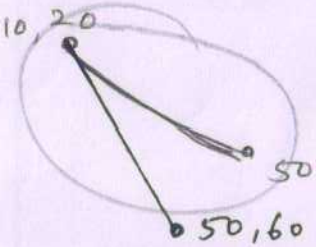
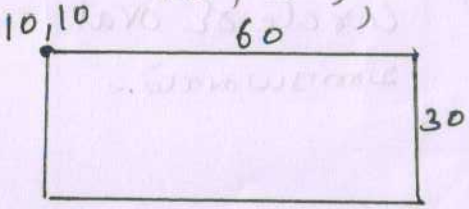
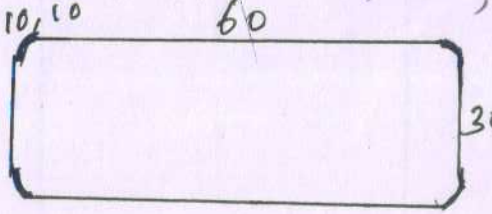
i) இது running stateன் மேற்கே எவை பயன்படுத்தலாம்.

ii) (a)

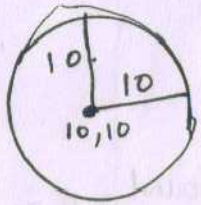
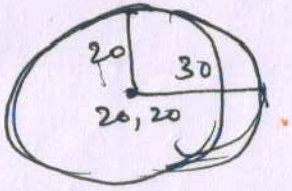
iii) (b).

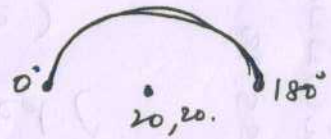
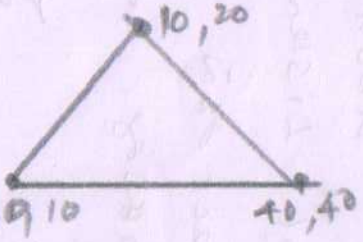
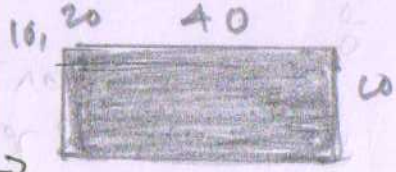
4) Explain about Graphics class.

ಈ class Java.awt ಪ್ಯಾಕೇಜ್ ನಲ್ಲಿ ಇರುತ್ತದೆ. Graphical images ಗಳನ್ನು ರಚಿಸಲು ಮತ್ತು ಬಣ್ಣಗಳನ್ನು ತುಂಬಲು ಈ class ನಲ್ಲಿ ಅನೇಕ methods ಗಳು ಲಭ್ಯವಿರುತ್ತವೆ.

method	use	syntax	Example
Line ①			Graphic g;
drawLine()	ಈ ಒಂದು method line ಅನ್ನು ರಚಿಸಲು ಬಳಸಲಾಗುತ್ತದೆ	drawLine(x1, y1, x2, y2); x1, y1 - starting point x2, y2 - end point.	g.drawLine(10, 20, 50, 60) 
Rectangle			
1. drawRect()	ಈ ಒಂದು method sharp corner rectangle ಅನ್ನು ರಚಿಸಲು ಬಳಸಲಾಗುತ್ತದೆ.	drawRect(x, y, w, h); x, y - starting point w - width of rectangle h - height of rectangle.	g.drawRect(10, 10, 60, 30); 
2. drawRound Rect()	ಈ ಒಂದು method round corner rectangle ಅನ್ನು ರಚಿಸಲು ಬಳಸಲಾಗುತ್ತದೆ.	drawRoundRect(x, y, w, h, ⑤ aw, ah); x, y - starting point w - width of rectangle h - height of rectangle. aw - arc width ah - arc height	g.drawRoundRect (10, 10, 60, 30, 5, 7); 

5

method	use	syntax	Example
3) draw3DRect()	இந்த method 3D rectangle வரைய பயன் படுகிறது.	draw3DRect(x, y, w, h, b); x, y - startingpoint w - width h - height b - either true or false.	g.drawRect(30, 30, 70, 20, 1);;
Circle drawOval()	இந்த methodஐ பயன் படுத்தி circle or oval வரையலாம்.	drawOval(x, y, w, h); x, y - starting point w - width h - height (w, h value same ஆக கொடுத்தால் circle வரையலாம்)	g.drawOval(20, 20, 40, 40);;  g.drawOval(20, 20, 30, 20); 

method	use	Syntax	Example
Arcs drawArc()	இந்த method பயன்படுத்தி arc வரையலாம்.	drawArc(x, y, w, h, sa, aa);	g.drawArc(20, 20, 40, 50, 0, 180); 
polygon drawPolygon()	இந்த method polygon வரைய பயன்படுகிறது.	drawPolygon(x[], y[], P) x[] – x coordinate values y[] – y coordinate values P – number of points.	g.drawpolygon(x1, y1, 3); x1[] = {40, 40, 10}; y1[] = {10, 40, 20}; 
Filling 1) fillRect() 2) fillRoundRect() 3) fill3DRect() 4) fillOval() 5) fillArc() 6) fillPolygon()	இந்த method image வரைய fill செய்ய பயன்படுகிறது.	syntax same as above except draw replaced by fill eg. fillRect(10, 20, 40, 10) →	

(5) Explain about event listener.

Event Listener என்பது ஒரு object. இது source object இல் generate ஆகின்ற event ஐ monitor ஆகவும் அதற்கான process ஐ ஆற்றவும் ஆகும்.

Name	use	methods

1. ActionListener	Button press ஆகின்றபோது (அ) menu item (அ) list item select ஆகின்றபோது generate ஆகும் event ஐ listen ஆகவும் process ஆகின்றது.	void actionPerformed (ActionEvent e)
		{
		{
		}
		}

2. AdjustmentListener	scrollbar move ஆகின்றது generate ஆகும் event ஐ listen ஆகவும் Process ஆகின்றது.	void adjustmentValueChanged (AdjustmentEvent e)
		{
		}
		}

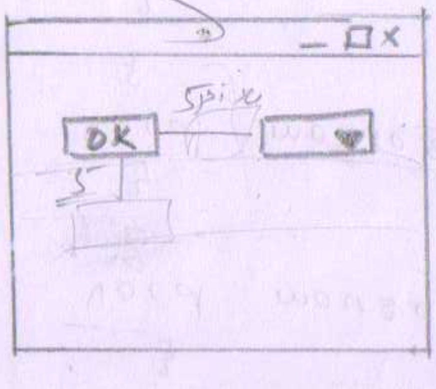
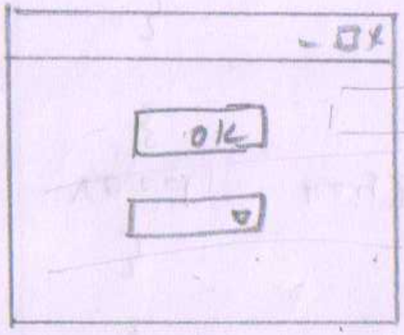
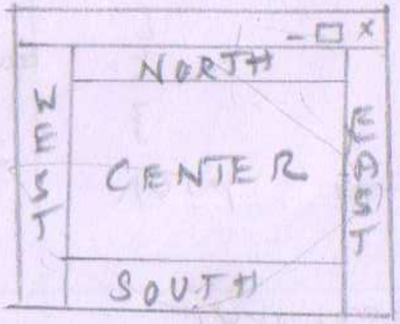
3. ItemListener	checkbox அ choice item click ஆகின்றபோது generate ஆகும் event ஐ listen ஆகவும் process ஆகின்றது.	void itemStateChanged (
		ItemEvent e)
		{
		}
		}

Name	use	methods
4. KeyListener	Keyboard ನಿಂದ type (ಅ) release (or) KeyPress ಆದಾಗ generate ಆದ event ಅನ್ನು listen ಮಾಡಿ process ಮಾಡುತ್ತದೆ.	void keyPressed(KeyEvent e) { . . . . } void keyReleased(KeyEvent e) { . . . . } void keyTyped(KeyEvent e) { . . . . }
5. MouseListener	ಯಾವ mouse ನ click (ಅ) release (ಅ) component ಗೆ enter (ಅ) exit ಆಗಿರುವಾಗ generate ಆದ event ಅನ್ನು listen ಮಾಡಿ process ಮಾಡುತ್ತದೆ.	void mouseClicked(MouseEvent e) { . . . . } void mousePressed(MouseEvent e) { . . . . } void mouseReleased(MouseEvent e) { . . . . } void mouseEntered(MouseEvent e) { . . . . } void mouseExited(MouseEvent e) { . . . .

6) Explain about Layout manager.

Windowsஇல் GUI controlஐ arrange செய்ய Layout manager பயன்படுகிறது. இதன் வகைகள் Layout managers ஆவன:

1. FlowLayout 2. BorderLayout 3. GridLayout.

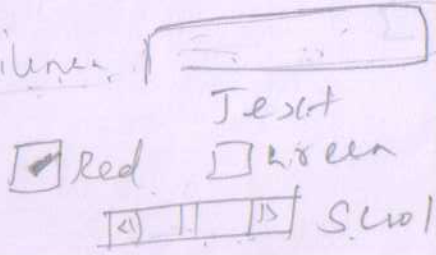
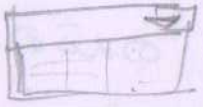
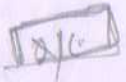
Name & use	Constructor	Example
1. FlowLayout (இந்த controlஐ left to right ஆக place செய்கிறது) controlகளுக்கு இடையே 5 pixel அளவு இடைவெளி காணப்படும்)	① FlowLayout (alignment) alignment – FlowLayout.CENTER FlowLayout.LEFT FlowLayout.RIGHT ② FlowLayout (alignment, hgap, vgap)	
2. GridLayout (இந்த controlகள் row மற்றும் column ஆக place செய்கிறது)	① GridLayout ( ) இது 1 row 1 column என ஒன்றை உருவாக்கும். ② GridLayout (row, column)	
3. BorderLayout (இந்த controlகள் windowவின் edgeகளில் place செய்கிறது.	① BorderLayout ( ) இது controlகள் North, south, east, west மற்றும் center ல் place செய்கிறது.	

② BorderLayout(L).

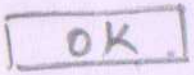
L may be

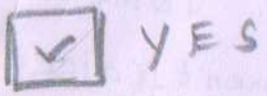
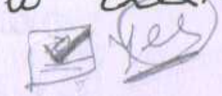
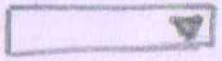
BorderLayout.CENTER
BorderLayout.NORTH
BorderLayout.SOUTH
BorderLayout.EAST
BorderLayout.WEST

⑦ Explain the different awt controls.

Name & use	Constructor	Methods	Events.
① Label ஒரு uneditable text ஐ display செய்ய. Control ஆகும்.  	1) Label() - blank label ஐ create செய்கிறது. 2) Label(Text) - text உடன் label ஐ create செய்கிறது. 3) Label(Text, a) - text ~~உடன்~~ குறிப்பிட்ட alignment உடன் create செய்கிறது. 	1) getText() - குறிப்பிட்ட label இன் text ஐ return செய்யும். 2) setText(S1) - S1 ஐ label இன் text ஆக set செய்கிறது.  - Button	-

2) TextField
single line textஐ display செய்வதற்கு அல்லது user இடமிருந்து பெறுவதற்குப் பயன்படுகிறது.
 
	1) TextField() – blank textfieldஐ create செய்கிறது.
	2) TextField(n) – n sizeல் textfieldஐ create செய்கிறது.
	3) TextField(S) – கொடுக்கப்பட்ட Sஐ display செய்து textfieldஐ create செய்கிறது.
		① getText() – தற்போது TextFieldல் உள்ள textஐ return செய்யும்.
		② setText(S) – கொடுக்கப்பட்ட Sஐ textfieldல் set செய்கிறது.
		③ getSelectedText() textfieldல் select செய்துள்ள textஐ return செய்யும்.
			public void actionPerformed(ActionEvent e)
			{
			. . .
			. . .
			}

3) Button push buttonஐ create செய்கிறது.  	1) Button() – blank buttonஐ create செய்கிறது 2) Button(S) – S உடன் கூடிய buttonஐ create செய்கிறது.	1) getLabel() – தற்போது buttonல் உள்ள labelஐ return செய்யும். 2) setLabel(S) – கொடுக்கப்பட்ட S buttonல் set செய்கிறது.	public void actionPerformed(ActionEvent e) { . . . . . . }

4) Checkbox optionஐ select (அ) deselect செய்ய உதவும் controlஐ create செய்கிறது.  	1) Checkbox() – blank Checkboxஐ create செய்கிறது.  2) Checkbox(s) – s உடன் checkboxஐ create செய்கிறது. 3) Checkbox(s, b) – s உடன் checkboxஐ create செய்கிறது. b = 1, checkbox selectஆகும் b = 0, checkbox deselectஆகும்	1) getLabel() – தற்போது checkboxன் உள்ள labelஐ return செய்யும். 2) setLabel(s) – sஐ checkbox labelஆக set செய்யும். 3) getState() – checkboxன் current state return செய்யும். 1, 0 4) setState(b) – checkboxன் stateஐ set செய்கிறது.	public void ~~itemSt~~ itemStateChanged(ItemEvent e) { . . . . . . }
5) Choice dropdown listஐ create செய்ய பயன்படுகிறது. 	1) Choice() – empty list create செய்கிறது.	1) add(text) – itemஐ choiceல் சேர்க்க உதவுகிறது. 2) getItemCount() – listல் உள்ள itemகளின் countஐ தருகிறது.	public void itemStateChanged(ItemEvent e) { . . . . . . }

6) Scrollbar
Vertical (or)
horintal scrollbar
create ஆகிய
பயன்படுத்து.

1) Scrollbar() - Vertical Scrollbar ஒன்று create ஆகும்.

2) Scrollbar(style) - குறிப்பிட்ட style உடன் scrollbar ஒன்று create ஆகும்.

style - Scrollbar.VERTICAL
Scrollbar.HORIZONTAL

3) select(n) - list-ன் nth item-ஐ select ஆகிய பயன்படுத்து.

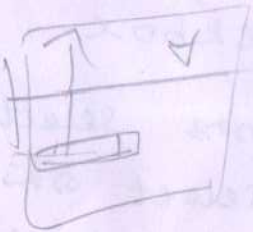
4) getItem(n) - list-ன் nth item-ஐ return ஆகிறது.

1) getValue() - scrollbar-ன் value-ஐ return ஆகும்

2) setValue(n) - scrollbar-ன் value-ஐ n ஆக set ஆகும்.

3) getMinimum() - Scrollbar-ன் minimum value-ஐ return ஆகும்.

4) getMaximum() - scrollbar-ன் maximum values return ஆகும்.



public void adjustmentValueChanged (AdjustmentEvent e)
{
. . . .
}

V unit

1. Explain about Exception Handling.

Program run ஆகும்போது ஏற்படும் errorகளை exception எனப்படும். Exceptionகளை கையாளும் techniqueகள் exception handling எனப்படும்.

Advantages

1. Runtimeன் போது அசாதாரணமான சூழல் ஏற்படாது பார்த்துக் கொள்கின்றது.
2. அசாதாரண சூழ்நிலை ஏற்பட்டதற்கான காரணத்தை தருகின்றது.

Exception types

Name	use
1. IOException	I/O operationனின் போது ஏற்படும் errorகளை அறிய பயன்படுகிறது.
2. ArithmeticException	Arithmetic operationனின் போது ஏற்படும் errorகளை அறிய பயன்படுகிறது.
3. ArrayIndexOutofBoundsException	Array index ஆனது தவறாக குறிப்பிடப்பட்டுள்ளதா என்பதை அறிய பயன்படுகிறது.
4. EOFException	End of File வந்துவிட்டதா என்பதை அறிய பயன்படுகிறது.
5) ClassNotFoundException	Class define செய்யப்பட்டுள்ளதா என்பதை அறிய பயன்படுகிறது.

Basics of Exception handling

```
try
  {
    statements for checking errors
  }
catch(Exceptionclass1  obj)
  {
    statements for handling the error
  }
catch (Exceptionclass2  obj)
  {
    statements for handling the error
  }
  .  .  .  .  .  .  .  .
  .  .  .  .  .  .  .  .
  .  .  .  .  .  .  .  .
catch(Exceptionclass n  obj)
  {
    statements for handling the error
  }
finally
  {
    statements
  }
```

try, catch, finally - Keywords.

try block

இந்த blockகுள் உள்ள Statementகளில் error உள்ளதா என்று சோதிக்கிறது. Error இருந்தால் அதை throw செய்கிறது. ஒரு Program ல் எவ்வளவு try blockகள் வேண்டுமானாலும்.

Catch block

இந்த block ஆனது try blockக்கு அடுத்ததாக வரவேண்டும். இது try blockல் இருந்து throw செய்யப்படுகின்ற exception objectகளை அதற்குரிய catch blockகள் catch செய்துகொள்ளும். ஒரு programல் ஒன்றுக்கும் மேற்பட்ட catch blockகள் வரலாம்.

Finally block

இந்த block optional ஆகும். Exception இருந்தாலும், இல்லாவிட்டாலும் இந்த block execute ஆகும். இதில் try - catch blockல் பயன்படுத்தப்பட்ட resourceகளை free செய்வதற்கான statementகள் கொடுக்கப்படும்.

eg

```
class Ex2
{
public static void main(String args[])
{
  int a = 25;
  int b = 0;
  int c;
  try
  {
    c = a/b;
   System.out.println("c = "+c);
  }
 catch (ArithmeticException e)
  {
   System.out.println("Divide by zero");
  }
```

```
finally
{
  System.out.println("Exception over");
}
```

2) Explain about defining and running a thread.

Thread ஐ இரண்டு முறைகளில் create செய்யலாம்.

1) creating threads by extending Thread class
2) creating threads by implementing Runnable interface.

Defining and running a thread using Thread class.

1) Thread class ஐ extend செய்து subclasses உருவாக்க வேண்டும்.

```
Class subclassname extends Thread
{
  .....
  .....
}
```

2) எல்லா subclass களிலும் run() method ஐ override செய்ய வேண்டும்.

```
public void run()
{
  Statements
}
```

3) main() method இல் thread objects ஐ create செய்ய வேண்டும்

4) thread object ஐ பயன்படுத்தி start()

Call ஆகும் போது.

```
threadobject.start();
```

eg.

```
class Even extends Thread
{
  public void run()
  {
    for(int i=0; i<=10; i=i+2)
      System.out.println("Even number:"+i);
  }
}
class Odd extends Thread
{
  public void run()
  {
    for(int j=1; j<=10; j=j+2)
      System.out.println("odd number:"+j);
  }
}
class T1
{
  public static void main(String args[])
  {
    Even e1 = new Even();
    Odd o1 = new Odd();
    e1.start();
    o1.start();
  }
}
```

Defining and running a thread using Runnable interface.

1) Thread subclassஐ Runnable interfaceஐ implement செய்து உருவாக்க வேண்டும்.

```
class subclassname implements Runnable
{
  . . . .
  . . . .
}
```

2) எல்லா subclassகளிலும் run() methodஐ override செய்ய வேண்டும்.

```
public void run()
{
  . . . . .
  . . . . .
}
```

3) main() methodல் thread objectஐ create செய்ய வேண்டும்.

4) Thread objectஐ பயன்படுத்தி start() call செய்ய வேண்டும்.

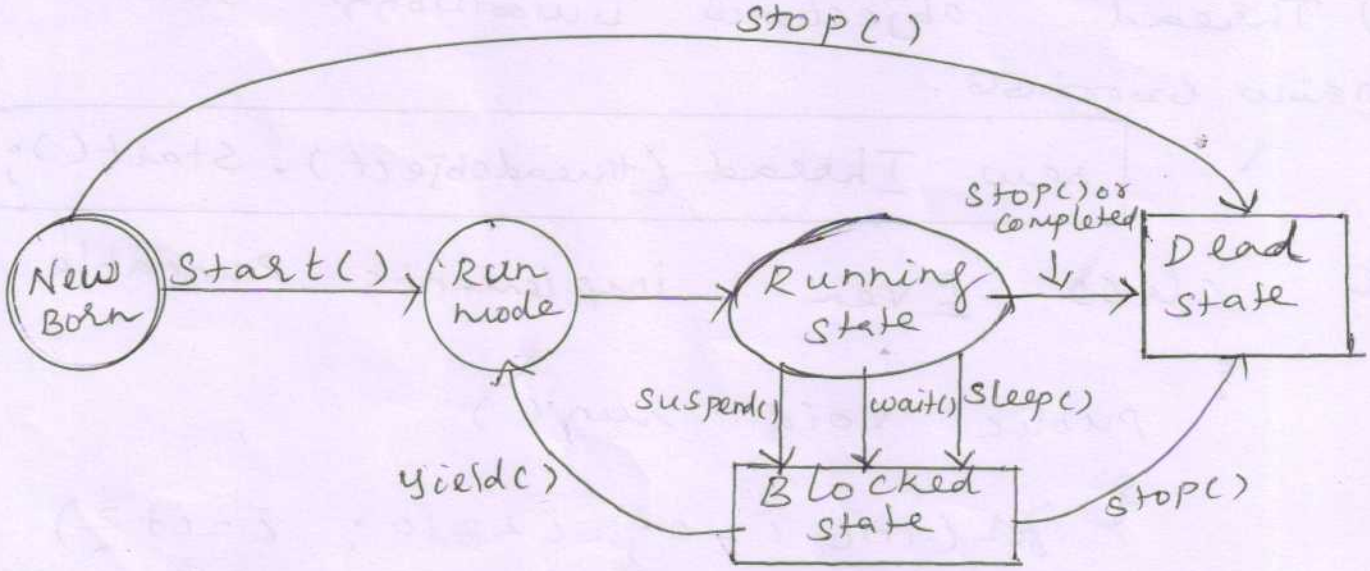
```
new Thread (threadobject). start();
```

eg:

```
class Even implements Runnable
{
  public void run()
  {
    for(int i=0; i<=10; i=i+2)
      System.out.println("Even no:"+i);
  }
}
class Odd implements Runnable
{
```

```
public void run()
{
  for (int j=1; j<=10; j=j+2)
    System.out.println("odd no:" + j);
}
}
Class T2
{
public static void main (String args[])
{
  Even e1 = new Even();
  Odd o1 = new Odd();
  new Thread(e1).start();
  new Thread(o1).start();
}
}
```

③ Explain the life cycle of Thread with neat diagram



Thread life cycle in 5 states in general.

1) Newborn state 2) Runnode state
3) Running state 4) Blocked state 5) Dead state

1) New born state

Define செய்த thread-க்கு thread object create செய்யும் போது புதிய thread ஆகிறது. இந்த state new born state எனப்படும். இந்த state வந்த thread runmode state அல்லது dead state-க்கு செல்லும்.

start() என்ற call செய்தால் runmode state-க்கு செல்லும். stop() என்ற call செய்தால் dead state-க்கு செல்லும்.

2) Runmode state.

Thread ஆனது execution-க்கு ready ஆகும் போது, runmode state என்று அழைக்கப்படுகிறது. Processor available ஆக இருக்கும் போது thread running state எனப்படும்.

3) Running state.

Thread execute ஆகும் போது running state என்கிறோம். இந்த state-ல் கண்ட ஏதேனும் ஒன்று நடக்கும் வரை continue ஆகிறது.

1) execution முடிந்து விட்டால் 2) Yield(), sleep(), wait() அல்லது suspend() call செய்யும் போது.

4) Blocked state

sleep(), wait() அல்லது suspend() call செய்யும் போது, thread ஆனது running state-லிருந்து Blocked state-க்கு செல்கிறது.

5) Dead State: stop() call செய்யும்போது அல்லது

execution complete ஆகும் வரை, thread ஆனது dead state க்கு வருகிறது.

④ Explain Byte Stream classes.

Byte Stream classகள், Byte I/O operation செய்ய பயன்படுகிறது. இரண்டு வகையான classகள் உள்ளன.

1) InputStream class 2) outputStream class.

InputStream class.

இது ஒரு abstract class ஆகும். இதில் input operationக்கு தேவையான methodகள் உள்ளன. Error ஏற்படும் IOException throw செய்யும்.

method	use.
1) read()	இந்த method input Streamலிருந்து ஒரு byteஐ read செய்ய பயன்படுகிறது.
2) read(byte b[])	இந்த method input Streamலிருந்து array of bytesஐ read செய்ய பயன்படுகிறது.
3) read(byte b[], n, m)	இந்த method i/p Streamலிருந்து nth positionலிருந்து m bytesஐ read செய்கிறது.
4) skip(n)	இந்த method n bytesஐ ~~skip~~ skip செய்ய பயன்படுகிறது

5) close()	இந்த method input stream ஐ close செய்ய பயன்படுகிறது.
6) reset()	இந்த method stream ன் தொடக்கத்திற்கு சென்று பயன்படுகிறது.
7) available()	இந்த method read செய்வதற்கு எத்தனை bytes கள் உள்ளன என்பதை தருகிறது.

outputstream class.

இது ஒரு abstract class. இதில் output operation க்கு தேவையான methods கள் உள்ளன. Error ஏற்பட்டால் IOException ஐ throw செய்யும்.

method	use.
1) write(b)	இந்த method ஒரு byte ஐ output stream ல் write செய்கிறது.
2) write(byte b[])	இந்த method array of bytes களை o/p stream ல் write செய்கிறது.
3) write(byte b[], n, m)	இந்த method nth position ல் இருந்து m bytes களை o/p stream ல் write செய்கிறது.
4) close()	இந்த method o/p stream ஐ close செய்கிறது.
5) flush()	இந்த method தான output buffer ஐ clear செய்கிறது.

(5) Explain characterstream classes.

Character stream classகள் characterல் I/O operation செய்ய பயன்படுகிறது. இரண்டு வகையான classகள் உள்ளன.

1) Reader 2) writer

(6) Explain about thread methods

method	use
1) start()	இந்த method run() methodஐ start செய்ய பயன்படுகிறது.
2) stop()	இந்த method running threadஐ stop செய்ய பயன்படுகிறது.
3) sleep(a)	இந்த method run ஆகும் threadஐ block செய்ய பயன்படுகிறது.
4) wait()	இந்த method run ஆகும் threadஐ stop செய்ய பயன்படுகிறது.
5) yield()	இந்த method blocked modeல் இருக்கும் threadஐ run modeக்கு மாற்ற பயன்படுகிறது.
6) isAlive()	இந்த method thread run ஆகிறதா என்பதை check செய்ய பயன்படுகிறது.

7) run()
public void run()
{
statements
}
	இந்த method threadகளை statementகளை கொண்டது.