UNIT II
PART A

Multiple Choice Questions

1. Which of the following statements will result in an infinite loop?
A. for(;;);
B. for(i=0;i<1;i--)
C. for(i=0;;i++)
D. **all the above**

2. In Java_____can only test for equality, whereas_____can evaluate any type of Boolean expression.
A. switch, if
B. if, switch
C. switch, break;
D. if, continue

3. Which of the following is an operator used for conditional branching in Java?
A. {}
B. ?:
C. >[]
D. <()

4. Which of the following loop is used when the number of iterations are known?
A. while
B. if
C. for
D. do-while

5. Which of the following is an exit controlled loop?
A. while
B. do-while
C. switch
D. for

6. Which of the following is an entry controlled loop?
A. do-while
B. switch
C. if
D. while

7. Name the built-in multi-way decision statement in Java.
A. if
B. if-else
C. switch
D. do-while

8. An early exit from a loop can be accomplished by using the_____statement in Java
A. continue
B. break
C. switch
D. goto

9. Which statement in Java causes the loop to be continued with the next iteration after skipping any statements in between?
A. break;
B. goto n;
C. continue;
D. switch

10. The data items in Java are known as _____

A. variables
B. methods
C. functions
D. fields

Questions based on Skill

11. The_____acts as a template for an object
A. class
B. state
C. behavior
D. field

12. A_____in Java is a user-defined data type with a template that serves to define its properties.
A. object
B. class
C. method
D. function

13. The _____ keyword in Java is used to implement inheritance
A. implements
B. extends
C. runnable
D. D. catch

14. The variables that are declared inside a class are called _____
A. instance variables
B. class variables
C. object members
D. D. object variables

15. Instance variables are also known as _____
A. member variables
B. class variables
C. object members
D. D. object variables

16. _____are also known as member variables in Java.
A. instance variables
B. class variables
C. object members
D. D. object variables

17. Any method declaration in a Java class consists of_____parts
A. two
B. three
C. four
D. D. five

18. The_____keyword in a method declaration means that the method does not return anything.
A. private
B. public
C. void
D. D. int

19. The parameter list in a method are enclosed in _____
A. []
B. {
}
C
.
(
)
D
.
<
>

20. Objects in Java are created using the operator operator in Java is used to access the class members

A. {}
B. ()
C. <u>New</u>
D. D. add

21. The _____

A. ?
B. ->
C. <u>dot</u>
D. new

22. The_____are used to give initial values to members in a class in Java

A**. constructor**

B. friend

C. finally

D. Runnable

23. In Java it is possible to create methods that have the same name, but different parameter lists and different definitions. This concept is called _____

A. inheritance

B. overloading

C. Encapsulation

D. D. Abstraction

24. _____are methods in Java that can be called without using the objects.

A. static methods

B. overloaded methods

C. constructors

D. D. destructors

25. A string is generally a sequence of characters. But, in Java, it is treated as _____

A. buffer

B. object

C. class

D. Package

Long Answer Question

1. Explain simple if statement in Java with an example. The general form of a simple if statement is

```
if(test expression)
{
    statement-block;
}
statement-x;
```

The 'statement-block' may be a single statement or a group of statements. If the *test expression* is true, the *statement-block* will be executed; otherwise the statement-block will be skipped and the execution will jump to the *statement-x.* It should be remembered that when the condition is true both the statementblock and the statement-x are executed in sequence.

Consider a case having two test conditions, one for weight and another for height. This is done using the compound relation if (weight < 50 && height> 170) count = count +1; This would have

been equivalently done using two if statements as follows: **if**(weight<50) **if** (height>170) count =count+l;

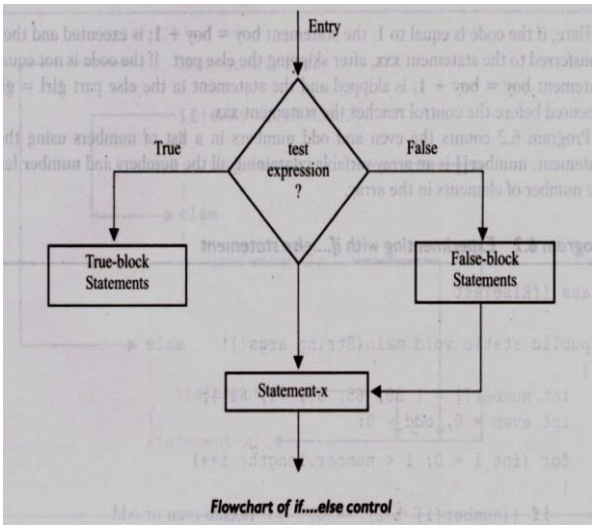
If the value of weight is less than **50,** then the following statement is executed. Which in turn is another if statement. This if statement tests height and if the height is greater than **170,** then the count is incremented by 1

2. How would you Explain if-else statement in Java with an example.

The if else statement is an extension of the simple if statement. The general form is

```
if(test expression)
{
   True-block statement(s)
}
else
{
   False-block statement(s)
}
statement-x
```

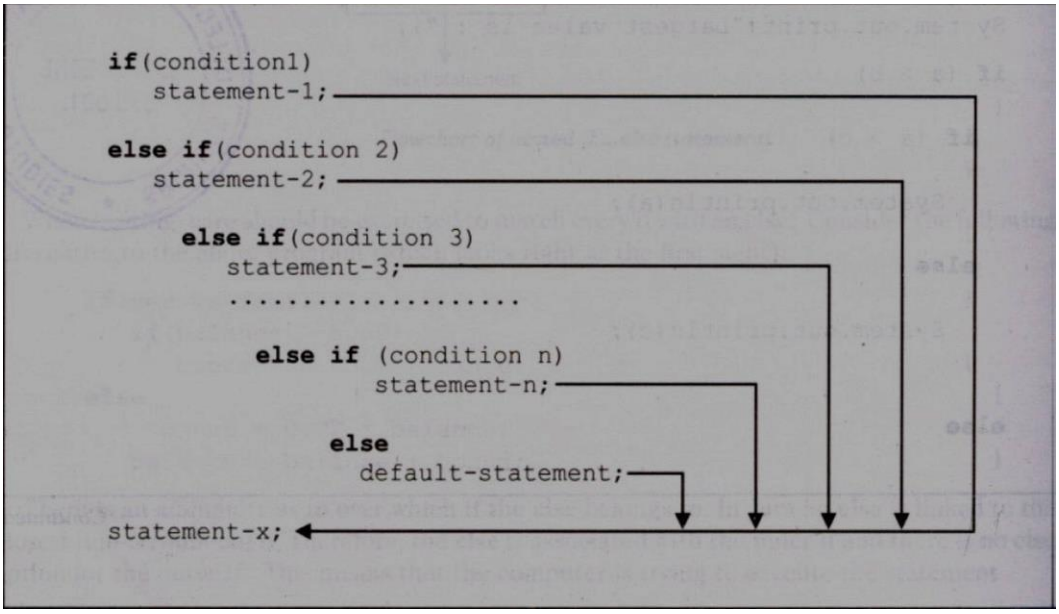
If the *test expression* is true, then *the true-block statement(s)* immediately following the if statement, are executed; otherwise, the *false-block statement(s)* are executed. In either case, either *true-block* or *false-block* will be executed, not both. In both the cases, the control is transferred subsequently to the *statement-x.* **if**(code == 1) boy = boy + 1; **else** girl = girl + 1;
XXX;



Flowchart of if....else control

3. Illustrate the usage of / else if ladder in Java with an example.

There is another way of putting ifs together when multipath decisions are involved. A multipath decision is a chain of ifs in which the statement associated with each else is an if. It takes the following general form



The construct is known as the else if ladder. The conditions are evaluated from the top *(of* the ladder), downwards. As *soon* as the true condition is found, the statement associated with it is executed and the control is transferred *to* the *statement-x* (skipping the rest *of* the ladder).when all the n conditions become false, then the final else containing the *default-statement* will be executed.

Consider an example *of* grading the students in an academic institution. The grading is done according to the following rules:

Average marks Grade 80 to 100 Honours

60 to 79 First Division SOto 59 Second Division 40 to 49 Third Division 0 to 39

Fail

This grading can be done using the else ***if*** ladder as follows~

if(marks>-79) grade = "Honours"; **else** if (marks >59) grade = "First Division"; **else**if (marks > 49) grade = "Second Division"; **else**if (marks > 39) grade = "Third Division"; **else** grade = "Fail"; System.out.println("Grade: " + grade);

4.How is the switch statement similar to else-if statement in Java? Explain with an example.

The general form of the switch statement is as shown below:

```
switch(expression)
{
      case value-1:
                block-1
                break;
      case value-2:
                block-2
                break;
      ......
      ......
      default:
                default-block
                break;
}
statement-x;
```

When the switch is executed, the value of the expression is successively compared against the values *value-l, value-2,* If a case is found whose value matches with the value of the expression, then the block of statements that follows the case are executed. The break statement at the end of each block signals the end of a particular case and causes an exit from the switch statement Example:-

```
.........
.........
index = marks/10;
switch(index)
{
    case 10:
    case 9:
      case 8:
           grade = "Honours";
           break;
      case 7:
      case 6:
           grade = "First Division";
           break;
      case 5:
           grade = "Second Division";
           break;
      case 4:
           grade = "Third Division";
           break;
      default:
           grade = "Fail";
           break;
}
System.out.println(grade);
```

4. Explain the working of while loop in Java with an example.

The simplest of all the looping structures in Java is the while statement. The basic format of the while statement is

```
Initialization;
While (test condition)
{
   Body of the loop
}
```

The while is an *entry-controlled* loop statement. The *test condition* is evaluated and if the condition is true, then the body of the loop is executed. After execution of the body, the test condition is once again evaluated and if it is true, the body is executed once again. This process of repeated execution of the body continues until the test condition finally becomes false and the control is transferred out of the 1oop. On exit, the program continues with the statement immediately after the body of the loop. Example;-

```
.........
.........
sum = 0;
n = 1;

while(n <= 10)
{
      sum = sum + n * n;
      n = n+1;
}
System.out.println("Sum = "+ sum);
.........
.........
```

6.How would you explain the usage of a for loop in Java with an example.

The for loop is another *entry-controlled* loop that provides a more concise loop control structure. The general form of the for loop is

```
for (intialization ; test condition ; increment)
{
     Body of the loop
}
```

The execution of the for statement is as follows:

1. *Initialization* of the *control variables* is done first, using assignment statements such as i = 1 and count = o. The variables i and count are known as loop-control variables.
2. The value of the control variable is tested using the *test condition.*
3. When the body of the loop is executed, the control is transferred back to the for statement after evaluating the last statement in the loop. Example:-

```
for (x  = 0 ; x < = 9 ; x = x+1)
{
     System.out.println(x);
}
```

7. Explain the do-while loop in Java with an example.

The while loop construct makes a test condition before the loop is executed. Therefore, the body of the loop may not be executed at all if the condition is not satisfied at the very first attempt. On some occasions it might be necessary to execute the body of the loop before the test is performed. Such situations can be handled with the help of the do statement. This takes theform:

```
Initialization;
do
{
   Body of the loop
}
while (test condition)
```

Consider the example:-

```
.........
.........
i = 1;
sum = 0;

do
{
      sum = sum + i;
      i = i+2;
}
while(sum < 40 || i < 10);
.........
.........
```

7.Can you justify the usage of objects in Java with an example.

Anything we wish to represent in a Java program must be encapsulated in a class that defines the *state* and *behaviour*of the basic program components known as *objects*

An object in Java is essentially a block of memory that contains space to store all the instance variables. Creating an object is also referred to as instantiating an object.

Objects in Java are created using the new operator. The new operator creates an object of the specified class and returns a reference to that object. Here is an example of creating an object of type Rectangle. Rectangle rect1 / / declare rect1 = new Rectangle ( ) / / instantiate Rectangle rect1 = new Rectangle ( );

The method Rectangle ( ) is the default constructor of the class. We can create any number of objects of Rectangle.

```
Rectangle  rect1 = new  Rectangle(  );
Rectangle  rect2 = new  Rectangle(  );
```

Example:-

```
class Rectangle
{
  int length, width;          //  Declaration of variables

  void getData(int x, int y)  //  Definition of method
  {
    length = x;
    width  = y;
  }
  int rectArea()              //  Definition of another method
  {
    int area = length * width;
    return (area);
  }
}
class RectArea                //  Class with main method
{
  public static void main(String args[ ])
  {
    int area1,area2;
    Rectangle rect1 = new Rectangle();     //  Creating objects
    Rectangle rect2 = new Rectangle();

    rect1.length = 15;                     //  Accessing variables
    rect1.width  = 10;

    area1 = rect1.length * rect1.width;

    rect2.getData(20,12);                  //  Accessing methods
    area2 = rect2.rectArea();

    System.out.println("Area1 = " + area1);
    System.out.println("Area2 = " + area2);
  }
}
```

8. Can you illustrate the concept of constructors in Java.

All objects that are created must be given initial values. We can do these using two approaches. The first approach uses the dot operator to access the instance variables and then assigns values to them individually. It can be a tedious approach to initialize all the variables of all the objects.

The second approach takes the help of a method like getData to initialize each object individually using statements like, rect1.getData (15, lO);

It would be simpler and more concise to initialize an object when it is first created. Java supports a special type of method, called a constructor that enables an object to initialize itself when it is created.

Constructors have the same name as the class itself. They do not specify a return type, not even void. This is because they return the instance of the class itself.

Let us consider our Rectangle class again. We can now replace the getData method by a constructor method as shown below:

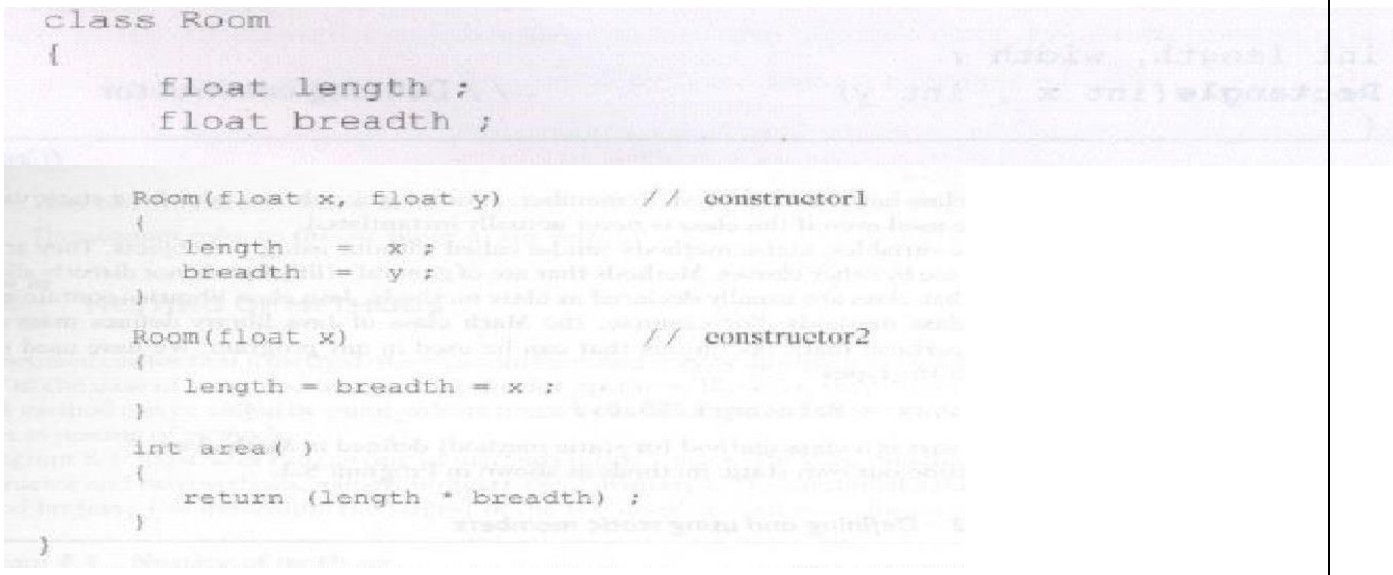
```
class Rectangle
{
      int length ;
      int width ;

      Rectangle(int x, int y) // Constructor method
      {
          length = x ;
          width  = y ;
      }
      int rectArea( )
      {
          return(length * width);
      }
}
```

```
class Rectangle
{
    int length, width ;
    Rectangle(int x , int y)                    // Defining constructor
    {
          length = x ;
          width  = y ;
    }
    int  rectArea( )
    {
        return (length * width);
    }
}
class RectangleArea
{
    public static void main(string args[ ])
  {
      Rectangle rect1 = new Rectangle(15,10); // Calling constructor
      int area1 = rect1.rectArea( ) ;
      System.out.println("Area1 = "+ area1) ;
   }
}
```

10. Explain the concept of method overloading in Java with an example.

In Java it is possible to create methods that have the same name, but different parameter lists and different definitions. This is called method overloading. Method overloading is used when objects are required to perform similar tasks but using different input parameters.

example of creating an overloaded method.



```
class Room
{
    float length ;
    float breadth ;

    Room(float x, float y)          // constructor1
    {
        length  =  x ;
        breadth  =  y ;
    }
    Room(float x)                   // constructor2
    {
        length = breadth = x ;
    }
    int area( )
    {
        return (length * breadth) ;
    }
}
```

Here, we are overloading the constructor method Room (). An object representing a rectangular room will be created as Room room1 = new Room (25.0, 15.0); //using constructor
On the other hand, if the room is square, then we may create the corresponding object as Room room2 = new Room (20.0); / / using constructor2

11. Can you explain the concept of static members in Java along with their limitations.

A class basically contains two sections. One declares variables and the other declares methods. These variables and methods are called instance variables and instance methods. This is because every time the class is instantiated, a new copy of each of them is created. They are accessed using the objects (with dot operator). Let us assume that we want to define a member that is common to all the objects and accessed without using a particular object. That is, the member belongs to the class as a whole rather than the objects created from the class. Such members can be defined as follows:
static int count;

static int max(intx, int y); ')

The members that are declared static as shown above are called static members.

Static variables are used when we want to have a variable common to all instances of a class

Like static variables, static methods can be called without using the objects.. For example, the Math class of Java library defines many static methods to perform math operations that can be used in any program. For example,

float x = Math.sqrt (25.0);

The method sqrt is a class method (or static method) defined in Math class.

Note that the static methods are called using class names. In fact, no objects have been created for use. Static methods have several restrictions:

1. They can only call other static methods.
2. They can only access static data.
3. They cannot refer to this or super in anyway

12. How can you explain the concept of working with classes in Java? Justify with an example.

A class is a user-defined data type with a template that serves to define its properties. Once the class type has been defined, we can create "variables" of that type. In Java, these variables are termed as *instances* of classes,

which are the actual *objects.* Classes provide a convenient method for packing together a group of logically related data items and functions that work on them. In Java, the data items are called fields and the functions are called *methods*

The basic form of a class definition is:

```
class   classname  [extends superclassname]
{
      [ variable declaration; ]
      [ methods declaration; ]
}
```

Everything inside the square brackets is optional. This means that the following would be a valid class definition:

```
class Empty
{
}
```

Example;-

```
class Rectangle
{
      int length;
      int width;

      void getData(int x , int y )
      {
         length  = x ;
         width   = y ;
      }
}
```

13. Compare and contrast while loop with do-while loop

While loop

The simplest of all the looping structures in Java is the while statement. The basic format of the while statement is

```
Initialization;
While (test condition)
{
   Body of the loop
}
```

The while is an *entry-controlled* loop statement. The *test condition* is evaluated and if the condition is true, then the body of the loop is executed. After execution of the body, the test condition is once again evaluated and if it is

true, the body is executed once again. This process of repeated execution of the body continues until the test condition finally becomes false and the control is transferred out of the 1oop. On exit, the program continues with the statement immediately after the body of the loop. Example;-

```
.....
.....
sum = 0;
n = 1;

while(n <= 10)
{
      sum = sum + n * n;
      n = n+1;
}
System.out.println("Sum = "+ sum);
.....
.....
```

Do-while

The while loop construct makes a test condition before the loop is executed. Therefore, the body of the loop may not be executed at all if the condition is not satisfied at the very first attempt. On some occasions it might be necessary to execute the body of the loop before the test is performed. Such situations can be handled with the help of the do statement. This takes theform:

```
Initialization;
do
{
   Body of the loop
}
while (test condition)
```

Example:-

```
.....
.....
i = 1;
sum = 0;

do
{
      sum = sum + i;
      i = i+2;
}
while(sum < 40 || i < 10);
.....
.....
```