


INSTITUTE OF TECHNOLOGY		CMR						USN							
Internal Assessment Test - I															
Sub:		Programming Using Java						Code:		18MCA21					
Date:		15.04.2019		Duration:		90 mins	Max Marks:		50	Sem:	II	Branch:		MCA	
Answer Any One FULL Question from each part.															
										Mar	OBE				
										ks	CO	RBT			
Part - I															
1	(a)	What are static methods and static block? Write a program to illustrate static method and static block?								[10]	CO1	L2			
2	(a)	Define method overloading and constructor overloading? Explain with example.								[10]	CO1	L2			
Part – II															
3	(a)	What is ‘this’ keyword? Demonstrate ‘this’ with a suitable program.								[5]	CO1	L2			
	(b)	What are instance and static variables? Explain with a suitable program.								[5]	CO1	L2			
4	(a)	Explain garbage collection and finalizers in java?								[6]	CO1	L2			
	(b)	What is for-each loop? Write its syntax.								[4]	CO1	L1			
PART - III															
5	(a)	In how many ways can we create a string in java? Why strings are immutable?								[5]	CO1	L2			
	(b)	Explain the difference between String, String Buffer and String Builder class.								[5]	CO1	L2			
6	(a)	What is toString(). Explain with example how to override toString().								[10]	CO1	L2			
Part – IV															
7	(a)	Explain following method of String class with example: indexOf(), lastIndexOf(),charAt(), getChars()								[10]	CO1	L2			
8	(a)	Write a program in Java for String handling which performs the following: i. Checks the capacity of StringBuffer objects. ii. Reverses the contents of a string given on console and converts the resultant string in upper case. iii. Reads a string from console and appends it to the resultant string of ii.								[10]	CO1	L3			
Part – V															
9	(a)	Explain four object oriented features of Java.								[4]	CO1	L2			
	(b)	What is inner class? Write a program to demonstrate inner class.								[6]	CO1	L1			
10	(a)	Briefly explain varargs with example program.								[5]	CO1	L2			
	(b)	What is Irregular Arrays? Explain with a example program.								[5]	CO1	L2			

1 a. What are static methods and static block? Write a program to illustrate static method and static block?

Ans: Static Method:

Any method which is declared with static keyword, it is known as static method.

- A static method belongs to the class rather than object of a class.
- A static method can be invoked without the need for creating an instance of a class.
- static method can access static data member and can change the value of it.

Static method example

```
class Student9{
    int rollno;
    String name;
    static String college = "ITS";
    static void change(){
        college = "BBDIT";
    }

    Student9(int r, String n){
        rollno = r;
        name = n;
    }

    void display () {System.out.println(rollno+" "+name+" "+college);
}
}
```

Class Demo

```
{
    static
    {
        System.out.println("Static block invoked");

        public static void main(String args[])
        {
            Student9.change();
            Student9 s1 = new Student9 (111,"Karan");
            Student9 s2 = new Student9 (222,"Aryan");
            Student9 s3 = new Student9 (333,"Sonoo");

            s1.display();
            s2.display();
            s3.display();
        }
    }
}
```

Java static block

- Is used to initialize the static data member.
- It is executed before main method at the time of class loading.

Static Block Example:

```
class Test {
    static int i;
    int j;
    Test()
    {
        System.out.println("Constructor called ");
    }
}
```

```

}
// start of static block
static {
    i = 10;
    System.out.println("static block called ");
}
// end of static block
}

class Demo {
    public static void main(String args[]) {

        // Although we don't have an object of Test, static block is
        // called because i is being accessed in following statement.
        System.out.println(Test.i);
    }
}

```

Output:

```

static block called
Constructor called
10

```

2 a. Define method overloading and constructor overloading? Explain with example.

Ans: If a class have multiple methods by same name but different parameters, it is known as **Method Overloading**.

If we have to perform only one operation, having same name of the methods increases the readability of the program.

Method overloading **increases the readability of the program**.

There are two ways to overload the method in java

1. By changing number of arguments
2. By changing the data type

class Calculation

```

{
    void sum(int a,int b)
    {
        System.out.println(a+b);
    }
    void sum(int a,int b,int c)
    {
        System.out.println(a+b+c);
    }
}

```

Class MethodOverload

```

{
    public static void main(String args[])
    {
        Calculation obj=new Calculation();
        obj.sum(10,10,10);
        obj.sum(20,20);
    }
}

```

Constructor Overloading

Constructor overloading is a technique in Java in which a class can have any number of constructors that differ in parameter lists. The compiler differentiates these constructors by taking into account the number of parameters in the list and their type.

EX:

```
class Student{
    int id;
    String name;
    int age;
    Student(int i,String n)
{
    id = i;
    name = n;
}
    Student(int i,String n,int a)
{
    id = i;
    name = n;
    age=a;
}
    void display()
{
    System.out.println(id+" "+name+" "+age);
}

    public static void main(String args[]){
        Student s1 = new Student(111,"Karan");
        Student s2 = new Student(222,"Aryan",25);
        s1.display();
        s2.display();
    }
}
```

3a. What is 'this' keyword? Demonstrate 'this' with a suitable program.

Ans: The 'this' keyword is used for two purposes

1. It is used to point to the current active object.
2. Whenever the formal parameters and data members of a class are similar, to differentiate the data members of a class from formal arguments the data members of a class are preceded with 'this'.

This(): It is used for calling current class default constructor from current class parameterized constructor.

This(...): It is used for calling current class parameterized constructor from other category constructors of the same class.

Ex: class Test

```
{
    int a,b;
    Test()
    {
        This(10);
        System.out.println("I am from default constructor");
        a=1;
        b=2;
        System.out.println("Value of a="+a);
        System.out.println("Value of b="+b);
    }
    Test(int x)
    {
        This(100,200);
    }
}
```

```

        System.out.println("I am from parameterized constructor");
        a=b=x;
        System.out.println("Value of a="+a);
        System.out.println("Value of b="+b);
    }
    Test(int a,int b)
    {
        System.out.println("I am from double parameterized constructor")           this.a=a+5;
        This.b=b+5;
        System.out.println("Value of instance variable a="+this.a);
        System.out.println("Value of instance variable b="+this.b);
        System.out.println("Value of a="+a);
        System.out.println("Value of b="+b);
    }
}
Class TestDemo3
{
    Public static void main(String k[])
    {
        Test t1=new Test();
    }
}

```

3b. What are instance and static variables? Explain with a suitable program.

Ans: Class variables also known as static variables are declared with the *static* keyword in a class, but outside a method, constructor or a block.

- There would only be one copy of each class variable per class, regardless of how many objects are created from it.
- Static variables are rarely used other than being declared as constants. Constants are variables that are declared as public/private, final and static. Constant variables never change from their initial value.
- Static variables are stored in static memory. It is rare to use static variables other than declared final and used as either public or private constants.
- Static variables are created when the program starts and destroyed when the program stops.
- Visibility is similar to instance variables. However, most static variables are declared public since they must be available for users of the class.
- Default values are same as instance variables. For numbers, the default value is 0; for Booleans, it is false; and for object references, it is null. Values can be assigned during the declaration or within the constructor. Additionally values can be assigned in special static initializer blocks.
- Static variables can be accessed by calling with the class name as *ClassName.VariableName*.
- When declaring class variables as public static final, then variables names (constants) are all in upper case. If the static variables are not public and final the naming syntax is the same as instance and local variables.

Instance Variables:

- Instance variables are declared in a class, but outside a method, constructor or any block.

- When a space is allocated for an object in the heap, a slot for each instance variable value is created.
- Instance variables are created when an object is created with the use of the keyword 'new' and destroyed when the object is destroyed.
- Instance variables hold values that must be referenced by more than one method, constructor or block, or essential parts of an object's state that must be present throughout the class.
- Instance variables can be declared in class level before or after use.
- Access modifiers can be given for instance variables.
- The instance variables are visible for all methods, constructors and block in the class. Normally, it is recommended to make these variables private (access level). However visibility for subclasses can be given for these variables with the use of access modifiers.
- Instance variables have default values. For numbers the default value is 0, for Booleans it is false and for object references it is null. Values can be assigned during the declaration or within the constructor.
- Instance variables can be accessed directly by calling the variable name inside the class. However within static methods and different class (when instance variables are given accessibility) should be called using the fully qualified name *.ObjectReference.VariableName*

Ex:

Class Sample

```
{
    int i,n;
    static int count=0;
    Sample(int a)
    {
        n=a;
    }
    Void even()
    {
        for(i=2;i<=n;i++)
        {
            If(i%2==0)
                Count++;
        }
    }
}
```

```
}
```

Class Demo

```
{
    Public static void main(String k[])
    {
        Sample s=new Sample(50);
        s.even();
        System.out.println(Sample.count+even numbers are present upto"+s.n);
    }
}
```

4a. Explain garbage collection and finalizers in java?

Ans: The technique of deallocating the memory automatically is called garbage collection. When an object

is no longer used then JVM automatically deletes it to free the memory

Before an object is destroyed, the resources linked to it should be freed. By using finalization, we can define specific actions that will occur when an object is just about to be reclaimed by the garbage collector.

The java run time calls that method whenever it is about to recycle an object of that class. Inside the **finalize()** method we will specify those actions that must be performed before an object is destroyed. The garbage collector runs periodically, checking for objects that are no longer referenced by any running state or indirectly through other referenced objects

The **finalize()** method has this general form:

```
protected void finalize()  
{  
    // finalization code here
```

4b. What is for-each loop? Write its syntax.

Ans: For-each loop is used to access the elements of an array or list. We can access the elements directly instead of index. The type of the loop variable and array type should be the same.

Syntax: for(type var:array)

```
{  
    Body of the loop;  
}
```

5a. In how many ways can we create a string in java? Why strings are immutable?

Ans: string is a sequence of characters. But in java, string is an object that represents a sequence of characters. String class is used to create string object.

There are two ways to create String object:

1. By string literal
2. By new keyword

string objects are immutable. Immutable simply means unmodifiable or unchangeable. Once string object is created its data or state can't be changed but a new string object is created.

```
Class Test{
```

```
Public static void main(Strin[] k)
```

```
{
```

```
String s="CMR";
```

```
s.concat("college");
```

```
System.out.println(s);
}
}
```

5b. Explain the difference between String, String Buffer and String Builder class.

Ans:

	<i>String</i>	<i>StringBuffer</i>	<i>StringBuilder</i>
Storage Area	Constant String Pool	Heap	Heap
Modifiable	No (immutable)	Yes(mutable)	Yes(mutable)
Thread Safe	Yes	Yes	No
Performance	Fast		Very
	slow	Fast	

6a. What is toString(). Explain with example how to override toString().

Ans:

Java toString() method

If you want to represent any object as a string, **toString() method** comes into existence.

The toString() method returns the string representation of the object.

If you print any object, java compiler internally invokes the toString() method on the object. So overriding the toString() method, returns the desired output, it can be the state of an object etc. depends on your implementation.

Advantage of Java toString() method

By overriding the toString() method of the Object class, we can return values of the object, so we don't need to write much code.

```
class Student{
    int rollno;
    String name;
    String city;

    Student(int rollno, String name, String city){
        this.rollno=rollno;
        this.name=name;
        this.city=city;
    }

    public String toString(){//overriding the toString() method
        return rollno+" "+name+" "+city;
    }
}
```



```

}
public static void main(String args[]){
    Student s1=new Student(101,"Raj","lucknow");
    Student s2=new Student(102,"Vijay","ghaziabad");

    System.out.println(s1);//compiler writes here s1.toString()
    System.out.println(s2);//compiler writes here s2.toString()
}
}

```

7a. Explain following method of String class with example:

indexOf(), lastIndexOf(),charAt(), getChars().

Ans: indexOf() : The **java string indexOf()** method returns index of given character value or substring. If it is not found, it returns -1. The index counter starts from zero.

```

Example: public class IndexOfExample{
public static void main(String args[]){
String s1="this is index of example";
int index1=s1.indexOf("is");//returns the index of is substring
int index2=s1.indexOf("index");//returns the index of index substring
System.out.println(index1+" "+index2);//2 8
    int index3=s1.indexOf("is",4);//returns the index of is substring after 4th index
System.out.println(index3);//5 i.e. the index of another is
    int index4=s1.indexOf('s');//returns the index of s char value
System.out.println(index4);//3
}}

```

lastIndexOf():The **java string lastIndexOf()** method returns last index of the given character value or substring. If it is not found, it returns -1. The index counter starts from zero.

Example:

```

public class LastIndexOfExample{
public static void main(String args[]){
String s1="this is index of example";//there are 2 's' characters in this sentence
int index1=s1.lastIndexOf('s');//returns last index of 's' char value
System.out.println(index1);//6
}}

```

charAt() : The **Java String charAt(int index)** method returns the character at the specified index in a string. The index value that we pass in this method should be between 0 and (length of string-1).

Example:

```

public class CharAtExample {
    public static void main(String args[]) {
        String str = "Welcome to string handling tutorial";
        //This will return the first char of the string
        char ch1 = str.charAt(0);

        //This will return the 6th char of the string
        char ch2 = str.charAt(5);

        //This will return the 12th char of the string
        char ch3 = str.charAt(11);

        //This will return the 21st char of the string
        char ch4 = str.charAt(20);

        System.out.println("Character at 0 index is: "+ch1);
        System.out.println("Character at 5th index is: "+ch2);
        System.out.println("Character at 11th index is: "+ch3);
        System.out.println("Character at 20th index is: "+ch4);
    }
}

```

Output:

```

Character at 0 index is: W
Character at 5th index is: m
Character at 11th index is: s
Character at 20th index is: n

```

getChars(): The method `getChars()` is used for copying `String` characters to an `Array` of chars.

```

public void getChars(int srcBegin, int srcEnd, char[] dest, int destBegin)

```

Parameters

description:

srcBegin – index of the first character in the string to copy.
srcEnd – index after the last character in the string to copy.
dest – Destination array of characters in which the characters from `String` gets copied.
destBegin – The index in `Array` starting from where the chars will be pushed into the `Array`.

Example:

```

public class GetCharsExample{
    public static void main(String args[]){
        String str = new String("This is a String Handling Tutorial");
        char[] array = new char[6];
        str.getChars(10, 16, array, 0);
        System.out.println("Array Content:");
        for(char temp: array){
            System.out.print(temp);
        }

        char[] array2 = new char[]{'a','a','a','a','a','a','a','a'};
        str.getChars(10, 16, array2, 2);
        System.out.println("Second Array Content:");
        for(char temp: array2){
            System.out.print(temp);
        }
    }
}

```

Output:

Array Content:
StringSecond Array Content:
aaString

8a. Write a program in Java for String handling which performs the following:

- i. Checks the capacity of StringBuffer objects.
- ii. Reverses the contents of a string given on console and converts the resultant string in upper case.
- iii. Reads a string from console and appends it to the resultant string of ii.

Ans: import java.util.Scanner;

```
public class stringclass {  
    public static void main(String args[])  
    {  
        StringBuffer sb=new StringBuffer("MCA");  
        System.out.println("length="+sb.length());  
        System.out.println("total capacity="+sb.capacity());  
        String s = new String("Aslam");  
        s=s.toUpperCase();  
        System.out.println(s);  
        String reverse =new StringBuffer(s).reverse().toString();  
        System.out.println("reversed string is"+reverse);  
        Scanner user = new Scanner(System.in);  
        System.out.println("enter any string");  
        String s3=user.next();  
        reverse=reverse.concat(s3);  
        System.out.println(reverse);  
    }  
}
```

9a. Explain four object oriented features of Java.

Ans: Ans: Inheritance

When one object acquires all the properties and behaviours of parent object i.e. known as inheritance. It provides code reusability. It is used to achieve runtime polymorphism.



Polymorphism

When one task is performed by different ways i.e. known as polymorphism. For example: to convince the customer differently, to draw something e.g. shape or rectangle etc.

In java, we use method overloading and method overriding to achieve polymorphism.

Another example can be to speak something e.g. cat speaks meaw, dog barks woof etc.

Abstraction

Hiding internal details and showing functionality is known as abstraction. For example: phone call, we don't know the internal processing.

In java, we use abstract class and interface to achieve abstraction.



Encapsulation

Binding (or wrapping) code and data together into a single unit is known as encapsulation. For example: capsule, it is wrapped with different medicines.

A java class is the example of encapsulation. Java bean is the fully encapsulated class because all the data members are private here.

9b. What is inner class? Write a program to demonstrate inner class.

Ans: In Java, just like methods, variables of a class too can have another class as its member. Writing a class within another is allowed in Java. The class written within is called the **nested class**, and the class that holds the inner class is called the **outer class**.

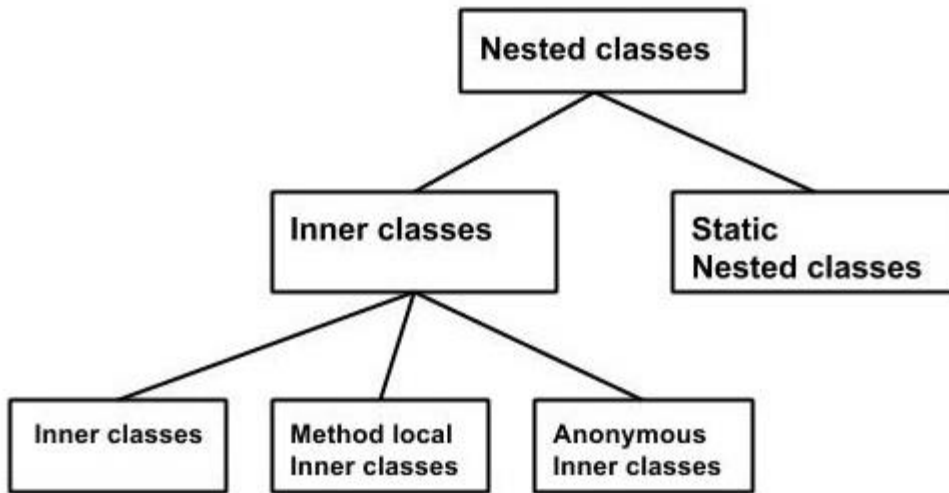
Syntax

Following is the syntax to write a nested class. Here, the class **Outer_Demo** is the outer class and the class **Inner_Demo** is the nested class.

```
class Outer_Demo {
    class Nested_Demo {
    }
}
```

Nested classes are divided into two types –

- **Non-static nested classes** – These are the non-static members of a class.
- **Static nested classes** – These are the static members of a class.



Inner Classes (Non-static Nested Classes)

Inner classes are a security mechanism in Java. We know a class cannot be associated with the access modifier **private**, but if we have the class as a member of other class, then the inner class can be made private. And this is also used to access the private members of a class.

Inner classes are of three types depending on how and where you define them. They are –

- Inner Class
- Method-local Inner Class
- Anonymous Inner Class

Inner Class

Creating an inner class is quite simple. You just need to write a class within a class. Unlike a class, an inner class can be private and once you declare an inner class private, it cannot be accessed from an object outside the class.

Following is the program to create an inner class and access it. In the given example, we make the inner class private and access the class through a method.

Example

```

class Outer_Demo {
    int num;

    // inner class
    private class Inner_Demo {
        public void print() {
            System.out.println("This is an inner class");
        }
    }
}

// Accessing the inner class from the method within
void display_Inner() {
    Inner_Demo inner = new Inner_Demo();
    inner.print();
}
  
```

```

}
}

public class My_class {

    public static void main(String args[]) {
        // Instantiating the outer class
        Outer_Demo outer = new Outer_Demo();

        // Accessing the display_Inner() method.
        outer.display_Inner();
    }
}

```

10a. Briefly explain varargs with example program.

Ans: Let's suppose you are creating a Java method. However, you are not sure how many arguments your method is going to accept. To address this problem, Java 1.5 introduced varargs.

args is a short name for variable arguments. In Java, an argument of a method can accept arbitrary number of values. This argument that can accept variable number of values is called varargs.

The syntax for implementing varargs is as follows:

```

accessModifier methodName(datatype... arg) {

    // method body

}

```

Example:

```

class NoVararg {

    public int sumNumber(int a, int b){
        return a+b;
    }

    public int sumNumber(int a, int b, int c){
        return a+b+c;
    }
}

```

```

public static void main( String[] args ) {
    NoVararg obj = new NoVararg();
    System.out.println(obj.sumNumber(1, 2));
    System.out.println(obj.sumNumber(1, 2, 3));
}
}

```

Output:

3
6

10b. What is Irregular Arrays? Explain with a example program.

Ans: Arrays that have elements of the same size are called rectangular arrays. It is possible to create irregular arrays where the arrays have a different size. In C# such arrays are called *jagged arrays*.

```

public class IrregularArrays {
    public static void main(String[] args) {
        int[][] ir = new int[][] {
            {1, 2},
            {1, 2, 3},
            {1, 2, 3, 4}
        };
        for (int[] a : ir) {
            for (int e : a) {
                System.out.print(e + " ");
            }
        }
        System.out.print('\n');
    }
}

```

Output: 1 2 1 2 3 1 2 3 4