CMR INSTITUTE OF TECHNOLOGY	USN										

Internal Assesment Test - ISub:

Sub	Programming with Java							Code:	16MCA32
Date:	18.09.2017	Duration:	90 mins	Max Marks:	50	Sem:	III	Branch:	MCA

Answer Any One FULL Question from each part.

		Marks	OBE	
			CO	RBT
Part I-1(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
2(a)	Define method overloading and constructor overloading? Can final methods be overridden .Justify your answer.	[6]	CO1	L1
(b)	What is method overriding? Write a program in java to illustrate it.	[4]	CO2	L1,L3
Part II-3(a)	How is multiple inheritance achieved in java? Write a program to implement multiple inheritance in java.	[5]	CO2	L2
(b)	Write the differences between abstract class and interface.	[5]	CO2	L2
4(a)	What is type casting? Explain the types of casting with an example program.	[4]	CO1	L1
(b)	Write a program to reverse the words of the string.	[6]	CO1	L3

Part III-5(a)	What are static methods and static block? Write a program to illustrate static method and static block?	[6]	CO1	L1,L3
(b)	In how many ways can we create a string in java? Why strings are immutable? Show the immutability with an example.	[4]	CO1	L3
6(a)	Explain garbage collection and finalizers in java?	[06]	CO1	L2
(b)	What is for-each loop? Write its syntax.	[04]	CO1	L1
Part IV-7(a)	Explain the difference between String, String Buffer and String Buider class.	[5]	CO1	L3
(b)	Explain all the methods to modify a String.	[5]	CO1	L3
8(a)	Explain bitwise operator.	[5]	CO1	L3
(b)	What is inner class? Write a program to demonstrate inner class.	[5]	CO1	L1,L3
Part V-9(a)	Explain how super class constructor is called using Super.	[5]	CO1	L3
(b)	Explain any three object oriented features of Java.	[5]	CO1	L3
10	Write a JAVA program which has i). A Interface class for Stack Operations ii). A Class that implements the Stack Interface and creates a fixed length Stack. iii).A Class that implements the Stack Interface and creates a Dynamic length Stack. iv). A Class that uses both the above Stacks through Interface reference and does the Stackoperations that demonstrates the runtime binding.	[10]	CO2	L3

Course Outcomes		PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8
CO1:	Understand the basic programming constructs of Java. Apply suitable OOP concepts to develop Java programs for a given scenario.	2	2	2	-	-	-	3	3
CO2:	Illustrate the concepts of Generalization and run time polymorphism applications	2	2	3	-	-	-	2	3
CO3:	Exemplify the usage of Packages, Interfaces, Exceptions and Multithreading	1	3	3	1	-	-	3	3
CO4:	Demonstrate Enumerations, Wrappers, Auto boxing, Generics, collection framework and I/O operations	1	2	3	-	-	-	3	3
CO5:	Implement the concepts of Networking using Java network classes	1	1	2	-	-	-	3	3

Cognitive level	KEYWORDS
L1	List, define, tell, describe, identify, show, label, collect, examine, tabulate, quote, name, who, when, where, etc.
L2	summarize, describe, interpret, contrast, predict, associate, distinguish, estimate, differentiate, discuss, extend
L3	Apply, demonstrate, calculate, complete, illustrate, show, solve, examine, modify, relate, change, classify, experiment, discover.
L4	Analyze, separate, order, explain, connect, classify, arrange, divide, compare, select, explain, infer.
L5	Assess, decide, rank, grade, test, measure, recommend, convince, select, judge, explain, discriminate, support, conclude, compare, summarize.

PO1 - Apply *knowledge*; PO2 - *Problem analysis*; PO3 - *Design/development of solutions*; PO4 - team work; PO5 - *Ethics*; PO6 - Communication; PO7- *Business Solution*; PO8 – Life-long learning;

CMR INSTITUTE OF TECHNOLOGY

Department of MCA

Odd Semester 2017

Internal Examination – I (Key)

Semester : III
Subject Code : 16MCA32
Subject Name : Programming with Java

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

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 preceeded 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();
        }
}
```

b) What are instance and static variables? Explain with a suitable program.
Static Variables:

- 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);
        }
}
```

2.a) What is method overloading and constructor overloading?can final methods be overridden? Justify your answer.

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();
  }
}
```

2(b) What is method overriding? Write a program in java to illustrate it.

If a sub class has the same method as declared in the parent class then it is called as method overriding.

Usage:

It is used to provide specific implementation of a method that is already provided by its super class used for run time polymorphism.

Rules:

1. Method must have same name and parameters as in parent class.
2. Method must be having is-a relation ship.

Ex:

```
class A
{
        int i,j;
        A(int a, int b)
        {
                i=a;
                j=b;
        }
        void show()
        {
                System.out.println("i and j value"+i+"    "+j);
        }
}
Class B  extends A
{
        int k;
        B(int a, int b,int c)
        {
                Super(a,b);
                k=c;
        }
        void show()
        {
                System.out.println("k  value"+k);
        }
}
Class Override
{
        Public static void main(String[] k)
        {
                B subob=new B(1,2,3);
                Subob.show();
        }
}
```

3(a) How is multiple inheritance achieved in java? Write a program to implement multiple inheritance in java.

Multiple Inheritance can be achieved through interfaces in java. As interfaces will contain only abstract methods it reduces ambiguity which causes when we perform through classes.

```
interface Printable
{
void print();
}

interface Showable
{
void show();
}
```

```
        class A7 implements Printable,Showable
        {

        public void print()
        {
                System.out.println("Hello");
       }
        public void show()
        {
                System.out.println("Welcome");
        }

        public static void main(String args[])
        {
                A7 obj = new A7();
                obj.print();
                obj.show();
         }
}
```

3(b) Write the differences between abstract class and interface.

Abstract class	Interface
1) Abstract class can **have abstract and non-abstract**methods.	Interface can have **only abstract** methods.
2) Abstract class **doesn't support multiple inheritance**.	Interface **supports multiple inheritance**.
3) Abstract class **can have final, non-final, static and non-static variables**.	Interface has **only static and final variables**.
4) Abstract class **can have static methods, main method and constructor**.	Interface **can't have static methods, main method or constructor**.
5) Abstract class **can provide the implementation of interface**.	Interface **can't provide the implementation of abstract class**.
6) The **abstract keyword** is used to declare abstract class.	The **interface keyword** is used to declare interface.

7) **Example:**
public abstract class Shape{
public abstract void draw();
}
	Example:
	public interface Drawable{
	void draw();
	}

4.a) What is type casting? Explain the types of casting with an example program.

Type casting is needed when we want to store a value of one type into a variable of another type.

There are two types of type casting .They are

1. Implicit Casting
2. Explicit Casting

Implicit Casting:

Automatic Type casting take place when,

- the two types are compatible

- the target type is larger than the source type

Narrowing or Explicit type conversion

When we are assigning a larger type value to a variable of smaller type, then we need to perform explicit type casting.

Example :public class Test{

Public static void main(String[] k)

{

Double d=100.87;

Long l=(long)d;

int i=(int)l;

System.out.println(“Double value”+d);

System.out.println(“Long value”+l);

System.out.println(“Int value”+i);

}

}

4 b) Write a program to reverse the words of the string.

Ans:

```
public class reverse {

public static void main(String[] args) {
	// TODO Auto-generated method stub
	String s1="This is Java";
	String [] s= s1.split(" ");
	String rev= "";
	for(int i=s.length-1;i>=0;i--)
	{
		rev+=s[i];
		rev+=" ";
	}
	System.out.println(rev);
}

}
```

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

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.

Java static block

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

Example of static method

```
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();
         }
      }
```

b) In how many ways can we create a string in java? Why strings are immutable? Show the immutability with an example.
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);

}

}

6)a)Explain garbage collection and finalizers in java?

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
}
```

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

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;
}
```

7(a) Explain the difference between String, String Buffer and String Buider class.

Ans:

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

7(b) Explain all the methods to modify a String.

Using substring()

replace()

trim()

8(a) Explain bitwise operator

Ans: Java defines several bitwise operators, which can be applied to the integer types, long, int, short, char, and byte.

Bitwise operator works on bits and performs bit-by-bit operation. Assume if a = 60 and b = 13; now in binary format they will be as follows −

a = 0011 1100

b = 0000 1101

---

a&b = 0000 1100

a|b = 0011 1101

a^b = 0011 0001

~a = 1100 0011

The following table lists the bitwise operators −

Assume integer variable A holds 60 and variable B holds 13 then −

Show Examples

Operator	Description	Example
& (bitwise and)	Binary AND Operator copies a bit to the result if it exists in both operands.	(A & B) will give 12 which is 0000 1100
| (bitwise or)	Binary OR Operator copies a bit if it exists in either operand.	(A | B) will give 61 which is 0011 1101
^ (bitwise XOR)	Binary XOR Operator copies the bit if it is set in one operand but not both.	(A ^ B) will give 49 which is 0011 0001
~ (bitwise compliment)	Binary Ones Complement Operator is unary and has the effect of 'flipping' bits.	(~A) will give -61 which is 1100 0011 in 2's complement form due to a signed binary number.

<< (left shift)	Binary Left Shift Operator. The left operands value is moved left by the number of bits specified by the right operand.	A << 2 will give 240 which is 1111 0000
>> (right shift)	Binary Right Shift Operator. The left operands value is moved right by the number of bits specified by the right operand.	A >> 2 will give 15 which is 1111
>>> (zero fill right shift)	Shift right zero fill operator. The left operands value is moved right by the number of bits specified by the right operand and shifted values are filled up with zeros.	A >>>2 will give 15 which is 0000 1111

8(b) what is inner class? Write a program to demonstrate inner class.

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**.

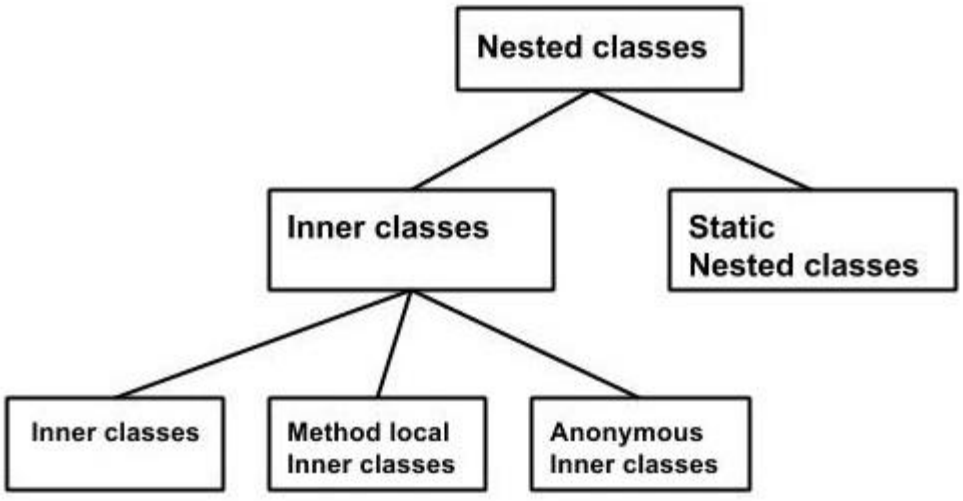
Syntax

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 he 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();
  }
}
```

9(a) Explain how super class constructor is called using Super.

Ans: The **super** keyword in java is a reference variable which is used to refer immediate parent class object.

Whenever you create the instance of subclass, an instance of parent class is created implicitly which is referred by super reference variable.

The super keyword can also be used to invoke the parent class constructor. Let's see a simple example:

```
1. class Animal{
2. Animal(){System.out.println("animal is created");}
3. }
4. class Dog extends Animal{
5. Dog(){
6. super();
7. System.out.println("dog is created");
8. }
9. }
10. class TestSuper3{
11. public static void main(String args[]){
12. Dog d=new Dog();
13. }}
```

Output:

```
animal is created
dog is created
```

9(b) Explain any three object oriented features of Java.

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.

---

10) Write a JAVA program which has
i). A Interface class for Stack Operations
ii). A Class that implements the Stack Interface and creates a fixed length Stack.
iii).A Class that implements the Stack Interface and creates a Dynamic length Stack.

iv). A Class that uses both the above Stacks through Interface reference and does the Stackoperations that demonstrates the runtime binding.

Ans: interface IntStack
{
	void push(int item);
	int pop();
}

class FixedStack implements IntStack
{
	private int stck[]; private int tos;
	FixedStack(int size)
	{
		stck=new int[size]; tos=-1;
	}
	public void push(int item)
	{
		if(tos==stck.length-1) System.out.println("Stack is Full\n");
		else
		stck[++tos]=item;
	}

	public int pop(){ if(tos<0){
		System.out.println("Stack underflow\n"); return 0;
		}
	else
	return stck[tos--];
	}
	}
class DynStack implements IntStack{ private int stck[];
	private int tos; DynStack(int size){
	stck=new int[size]; tos=-1;
}
public void push(int item)
{
	if(tos==stck.length-1)
	{

	int temp[]= new int[stck.length*2];
	  for(int i=0;i<stck.length;i++) temp[i]=stck[i]; stck=temp;
	  stck[++tos]=item;

	  }
	    else stck[++tos]=item;
}

public int pop()
{
   if(tos<0)
{
 System.out.println("Stack UnderFlow\n"); return 0;
}
else
 return stck[tos--];
}
}
class IFTest{
	public static void main(String args[])
{
	IntStack mystack;

```
        DynStack ds=new DynStack(5); FixedStack fs=new FixedStack(8); mystack=ds;

        for(int i=0;i<12; i++) mystack.push(i); mystack=fs;
        for(int i=0;i<8;i++) mystack.push(i); mystack=ds;
        System.out.println("Values in Dynamic Stack\n"); for(int i=0;i<12;i++) System.out.println(mystack.pop());
        mystack=fs;
        System.out.println("Values in fixed stack\n"); for(int i=0;i<8;i++)
      System.out.println(mystack.pop());
}
}
```