CMR INSTITUTE OF TECHNOLOGY

USN | | | | | | | | | | |

Internal Assesment Test - II

Sub: Programming Using Java Code: 18MCA21

Date: 13.05.2019 Duration: 90 mins Max Marks: 50 Sem: II Branch: MCA

Answer Any One FULL Question from each part.

			Marks	OBE CO	RBT
		Part - I			
1	(a)	Why final keyword is used. Explain the uses of final with variable, method and class.	[10]	CO2	L2
		OR			
2	(a)	Explain with example how overridden method support polymorphism.	[10]	CO2	L2
		Part – II			
3	(a)	What is an exception in java? Explain about compile time exceptions and run time exceptions.	[5]	CO3	L2
	(b)	Explain the usage of keywords try, throw, catch, finally, throws with their syntax.	[5]	CO3	L2
		OR			
4	(a)	What is inheritance? Explain the order of constructor execution in multilevel hierarchy of classes..	[6]	CO2	L2
	(b)	Explain access specifiers in java with an example.	[4]	CO3	L1

CMR INSTITUTE OF TECHNOLOGY

USN | | | | | | | | | | |

Internal Assesment Test - II

Sub: Programming Using Java Code: 18MCA21

Date: 13.05.2019 Duration: 90 mins Max Marks: 50 Sem: II Branch: MCA

Answer Any One FULL Question from each part.

			Marks	OBE CO	RBT
		Part - I			
1	(a)	Why final keyword is used. Explain the uses of final with variable, method and class.	[10]	CO2	L2
		OR			
2	(a)	Explain with example how overridden method support polymorphism.	[10]	CO2	L2
		Part – II			
3	(a)	What is an exception in java? Explain about compile time exceptions and run time exceptions.	[5]	CO3	L2
	(b)	Explain the usage of keywords try, throw, catch, finally, throws with their syntax.	[5]	CO3	L2
		OR			
4	(a)	What is inheritance? Explain the order of constructor execution in multilevel hierarchy of classes..	[6]	CO2	L2
	(b)	Explain access specifiers in java with an example.	[4]	CO3	L1

		PART - III			
5	(a)	What is Object class. List the methods defined in Object class.	[5]	CO2	L2

	(b)	Explain how super class constructor is called using Super.	[5]	CO2	L2
		OR			
6	(a)	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]	CO3	L3
		Part – IV			
7	(a)	Define a package. Explain the creation of package and create the package by the name shape to illustrate it.	[10]	CO3	L2
		OR			
8	(a)	. Differentiate between throw and throws. Explain finally with the help of example.	[10]	CO3	L4
		Part – V			
9	(a)	Write a java program to define multiple catch blocks.	[4]	CO3	L2
	(b)	**What is method overriding? Write a program in java to illustrate it**.	[6]	CO2	L1
		OR			
10	(a)	How is multiple inheritance achieved in java? Write a program to implement multiple inheritance in java.	[5]	CO2	L3
	(b)	Write the differences between abstract class and interface.	[5]	CO2	L2

PART - III

5	(a)	What is Object class. List the methods defined in Object class.	[5]	CO2	L2
	(b)	Explain how super class constructor is called using Super.	[5]	CO2	L2
		OR			
6	(a)	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]	CO3	L3
		Part – IV			
7	(a)	Define a package. Explain the creation of package and create the package by the name shape to illustrate it.	[10]	CO3	L2
		OR			
8	(a)	Differentiate between throw and throws. Explain finally with the help of example.	[10]	CO3	L4
		Part – V			
9	(a)	Write a java program to define multiple catch blocks.	[4]	CO3	L2
	(b)	**What is method overriding? Write a program in java to illustrate it**.	[6]	CO2	L1
		OR			
10	(a)	How is multiple inheritance achieved in java? Write a program to implement multiple inheritance in java.	[5]	CO2	L3
	(b)	Write the differences between abstract class and interface.	[5]	CO2	L2

1 a. Why final keyword is used. Explain the uses of final with variable, method and class.

Ans: final keyword in java

final keyword is used in different contexts. First of all, *final* is a non-access modifier applicable **only to a variable, a**

method or a class.Following are different contexts where final is used.

Final variables

When a variable is declared with *final* keyword, its value can't be modified, essentially, a constant. This also means that you must initialize a final variable. If the final variable is a reference, this means that the variable cannot be re-bound to reference another object, but internal state of the object pointed by that reference variable can be changed

```
class Gfg
{
   // a final variable
   // direct initialize
   final int THRESHOLD = 5;

   // a blank final variable
   final int CAPACITY;

   // another blank final variable
   final int  MINIMUM;

   // a final static variable PI
   // direct initialize
   static final double PI = 3.141592653589793;

   // a  blank final static  variable
   static final double EULERCONSTANT;

   // instance initializer block for
   // initializing CAPACITY
   {
      CAPACITY = 25;
   }

   // static initializer block for
   // initializing EULERCONSTANT
   static{
      EULERCONSTANT = 2.3;
   }

   // constructor for initializing MINIMUM
   // Note that if there are more than one
   // constructor, you must initialize MINIMUM
   // in them also
   public GFG()
   {
      MINIMUM = -1;
   }

}
```

Final classes

When a class is declared with *final* keyword, it is called a final class. A final class cannot be extended(inherited). There are two uses of a final class :

1. One is definitely to prevent inheritance, as final classes cannot be extended. For example, all Wrapper Classes like Integer ,Float etc. are final classes. We cannot extend them.

```
final class A
{
    // methods and fields
}
```

```
// The following class is illegal.
class B extends A
{
    // COMPILE-ERROR! Can't subclass A
}
```

Final methods

When a method is declared with *final* keyword, it is called a final method. A final method cannot be overridden. The Object class does this—a number of its methods are final.We must declare methods with final keyword for which we required to follow the same implementation throughout all the derived classes. The following fragment illustrates final keyword with a method:

```
class A
{
    final void m1()
    {
        System.out.println("This is a final method.");
    }
}

class B extends A
{
    void m1()
    {
        // COMPILE-ERROR! Can't override.
        System.out.println("Illegal!");
    }
}
```

2 a. Explain with example how overridden method support polymorphism.

Ans: In any object-oriented programming language, Overriding is a feature that allows a subclass or child class to provide a specific implementation of a method that is already provided by one of its super-classes or parent classes. When a method in a subclass has the same name, same parameters or signature and same return type(or sub-type) as a method in its super-class, then the method in the subclass is said to *override*the method in the super-class.
Method overriding is one of the way by which java achieve Polymorphism. The version of a method that is executed will be determined by the object that is used to invoke it. If an object of a parent class is used to invoke the method, then the version in the parent class will be executed, but if an object of the subclass is used to invoke the method, then the version in the child class will be executed.In other words, *it is the type of the object being referred to* (not the type of the reference variable) that determines which version of an overridden method will be executed.

```
class Parent
{
   void show() { System.out.println("Parent's show()"); }
}

// Inherited class
class Child extends Parent
{
   // This method overrides show() of Parent
   @Override
   void show() { System.out.println("Child's show()"); }
}

// Driver class
class Main
```

```
{
   public static void main(String[] args)
   {
      // If a Parent type reference refers
      // to a Parent object, then Parent's
      // show is called
      Parent obj1 = new Parent();
      obj1.show();

      // If a Parent type reference refers
      // to a Child object Child's show()
      // is called. This is called RUN TIME
      // POLYMORPHISM.
      Parent obj2 = new Child();
      obj2.show();
   }
}
```

Output:

```
Parent's show()

Child's show()
```

3 a What is an exception in java? Explain about compile time exceptions and run time exceptions.

Ans: An exception (or exceptional event) is a problem that arises during the**execution** of a program. When an Exception occurs the normal flow of the program is disrupted and the program/Application terminates abnormally, which is not recommended, therefore these exceptions are to be handled.

Checked exceptions: A checked exception is an exception that occurs at the compile time, these are also called as compile time exceptions. These exceptions cannot simply be ignored at the time of compilation, the Programmer should take care of (handle) these exceptions.

EX:

```
import java.io.File;
import java.io.FileReader;


public class FilenotFound_Demo {


   public static void main(String args[]){
      File file=new File("E://file.txt");
      FileReader fr = new FileReader(file);
   }


}
```

- **Unchecked exceptions:** An Unchecked exception is an exception that occurs at the time of execution, these are also called as Runtime Exceptions, these include programming bugs, such as logic errors or improper use of an API. runtime exceptions are ignored at the time of compilation.

  For example, if you have declared an array of size 5 in your program, and trying to call the 6th element of the array then an*ArrayIndexOutOfBoundsExceptionexception* occurs.

```
public class Unchecked_Demo {

  public static void main(String args[]){
    int num[]={1,2,3,4};
    System.out.println(num[5]);
  }

}
```

3 b. Explain the usage of keywords try, throw, catch, finally, throws with their syntax.

Ans: **Try:**

Try is used to guard a block of code in which exception may occur. This block of code is called guarded region

Syn:

try

{

Stmt-1;

}

Catch:

A catch statement involves declaring the type of exception you are trying to catch. If an exception occurs in guarded code, the catch block that follows the try is checked, if the type of exception that occured is listed in the catch block then the exception is handed over to the catch block which then handles it.

Syn:

Catch(ExceptionType obj)

{

Stmt

}

Throw:

Throw keyword is used to throw an exception explicitly. Only object of Throwable class or its sub classes can be thrown. Program execution stops on encountering throw statement, and the closest catch statement is checked for matching type of exception.

Syn:

Throw new throwableinstance;

Throws:

If a method does not handle a checked exception, the method must declare it using the throws keyword. The throws keyword appears at the end of a method's signature.

Syn:

```
type method_name(parameter_list) throws exception_list
{
 //definition of method
}
```

Finally:

A finally keyword is used to create a block of code that follows a try block. A finally block of code always executes whether or not exception has occurred. Using a finally block, lets you run any cleanup type statements that you want to execute, no matter what happens in the protected code. A finally block appears at the end of catch block.

Syn:

Finally

{

Block of statements

}

4 a. What is inheritance? Explain the order of constructor execution in multilevel hierarchy of classes.

Ans: Inheritance is a mechanism wherein a new class is derived from an existing class. In Java, classes may inherit or acquire the properties and methods of other classes.

A class derived from another class is called a subclass, whereas the class from which a subclass is derived is called a superclass. A subclass can have only one superclass, whereas a superclass may have one or more subclasses.

Order of execution of constructors in inheritance relationship is from base /parent class to derived / child class.

We know that when we create an object of a class then the constructors get called automatically.
In inheritance relationship, when we create an object of a child class, then first base class constructor and then derived class constructor get called implicitly.
If we create object of bottom most derived class i.e. of Testing class in main() program, then constructors of Design class, Coding class and then Testing class will be called.

```
class Design {
```

```
        Design(){

                System.out.println("Design()...");
        }

}

class Coding extends Design {

         Coding(){

                System.out.println("coding()...");
        }

}

class Testing extends Coding {

        Testing()
        {
```

```
        System.out.println("Testing()...");
        }

}

public class TestConstructorCallOrder {

        public static void main(String[] args) {

                //Create object of bottom most class object
                System.out.println("Constructor call order...");
                new Testing();

        }

}
```

OUTPUT:
Constructor call order…
Design()…
coding()…
Testing()…

4 b)Explain access specifiers in java with an example.

The access modifiers in java specifies accessibility (scope) of a data member, method, constructor or class.

There are 4 types of java access modifiers:

1. private
2. default
3. protected
4. public

Access Modifier	within class	within package	outside package by subclass only	outside package
Private	Y	N	N	N
Default	Y	Y	N	N
Protected	Y	Y	Y	N
Public	Y	Y	Y	Y

5 a. What is Object class. List the methods defined in Object class.

Ans: **Object** class is present in **java.lang** package. Every class in Java is directly or indirectly derived from the **Object** class. If a Class does not extend any other class then it is direct child class of **Object** and if extends other class then it is an indirectly derived. Therefore the Object class methods are available to all Java classes. Hence Object class acts as a root of inheritance hierarchy in any Java Program.

Using Object class methods

There are methods in **Object** class:

1. **toString()** : toString() provides String representation of an Object and used to convert an object to String.
2. **hashCode()** : For every object, JVM generates a unique number which is hashcode. It returns distinct integers for distinct objects.
3. **equals(Object obj)** : Compares the given object to "this" object (the object on which the method is called). It gives a generic way to compare objects for equality.
4. **getClass()** : Returns the class object of "this" object and used to get actual runtime class of the object.
5. **finalize()** method : This method is called just before an object is garbage collected. It is called by the Garbage Collector on an object when garbage collector determines that there are no more references to the object.
6. **clone()** : It returns a new object that is exactly the same as this object. For clone() method refer Clone()
7. The remaining three methods **wait()**, **notify() notifyAll()** are related to Concurrency.

5 b. 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. }}
```

Test it Now

Output:

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

6a. 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());
}
}
```

7a. Define a package. Explain the creation of package and create the package by the name shape to illustrate it.

Ans: A java package is a group of similar types of classes, interfaces and sub-packages.

Advantage of Java Package

1) Java package is used to categorize the classes and interfaces so that they can be easily maintained.

2) Java package provides access protection.

3) Java package removes naming collision.

Creation of Package:

While creating a package, we should choose a name for the package and include a package statement along with that name at the top of every source file that contains the classes, interfaces, enumerations, and annotation types that we want to include in the package.

The package statement should be the first line in the source file. There can be only one package statement in each source file, and it applies to all types in the file.

If a package statement is not used then the class, interfaces, enumerations, and annotation types will be placed in the current default package.

EX:

```
package shape;
public class circle {
   public void display1()
   {
       System.out.println("formula of circle is 3.142*r*r");
   }
}
package shape;
public class triangle {
   public void display3()
   {
      System.out.println("formula of triangle =0.5*length*breadth");
   }
}
package shape;
public class square {
   public void display2()
   {
       System.out.println("formula of Square =lengthv*width")
   }
 }
```

```
import shape.*;
public class prgrm7
{
   public static void main(String arg[])
   {
      circle ob1 = new circle();
      square ob2 =new square();
      triangle ob3 =new triangle();
      ob1.display1();
      ob2.display2();
      ob3.display3();
   }
}
```

8a. Differentiate between throw and throws. Explain finally with the help of example.

Ans:

Throw vs Throws

Throw	Throws
Java throw keyword is used to explicitly throw an exception.	Java throws keyword is used to declare an exception.
Checked exception cannot be propagated using throw only.	Checked exception can be propagated with throws.
Throw is followed by an instance.	Throws is followed by class.
Throw is used within the method.	Throws is used with the method signature.
You cannot throw multiple exceptions.	You can declare multiple exceptions e.g. public void method()throws IOException,SQLException.

Finally block:

Java finally block

Java finally block is a block that is used *to execute important code* such as closing connection, stream etc.

Java finally block is always executed whether exception is handled or not.

Java finally block follows try or catch block.

- Finally block in java can be used to put "cleanup" code such as closing a file, closing connection etc.

Syntax of Finally block

```
try {
    //Statements that may cause an exception
}
catch {
   //Handling exception
}
finally {
   //Statements to be executed
}
```

Example:

```
class Example
{
   public static void main(String args[]) {
      try{
                int num=121/0;
                System.out.println(num);
      }
      catch(ArithmeticException e){
         System.out.println("Number should not be divided by zero");
      }
      /* Finally block will always execute
       * even if there is no exception in try block
       */
      finally{
                System.out.println("This is finally block");
      }
      System.out.println("Out of try-catch-finally");
   }
}
```

Output:

```
Number should not be divided by zero
This is finally block
Out of try-catch-finally
```

9a. Write a java program to define multiple catch blocks.

Ans:

Java catch multiple exceptions

Java Multi-catch block

A try block can be followed by one or more catch blocks. Each catch block must contain a different exception handler. So, if you have to perform different tasks at the occurrence of different exceptions, use java multi-catch block.

Points to remember

- At a time only one exception occurs and at a time only one catch block is executed.

- All catch blocks must be ordered from most specific to most general, i.e. catch for ArithmeticException must come before catch for Exception.

Example 1

Let's see a simple example of java multi-catch block.

```
1.  public class MultipleCatchBlock1 {
2.
3.      public static void main(String[] args) {
4.
5.          try{
6.              int a[]=new int[5];
7.              a[5]=30/0;
8.             }
9.             catch(ArithmeticException e)
10.              {
11.               System.out.println("Arithmetic Exception occurs");
12.              }
13.            catch(ArrayIndexOutOfBoundsException e)
14.              {
15.               System.out.println("ArrayIndexOutOfBounds Exception occurs");
16.              }
17.            catch(Exception e)
18.              {
19.               System.out.println("Parent Exception occurs");
20.              }
21.            System.out.println("rest of the code");
22.    }
23. }
```

Output:

```
Arithmetic Exception occurs
rest of the code
```

9b. **What is method overriding? Write a program in java to illustrate it**.

Ans: 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();
	}
}
```

10 a. How is multiple inheritance achieved in java? Write a program to implement multiple inheritance in java.

Ans: 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();
         }
}
```

10 b. Write the differences between abstract class and interface.

Ans:

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();
	}