CMR
INSTITUTE OF
TECHNOLOGY

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

Internal Assessment Test II – August 2022

Sub:	Object Oriented Programming with Java							Sub Code:	20MCA22
Date:	29.08.22	Duration:	90 min's	Max Marks:	50	Sem:	II	Branch:	MCA

Note : Answer FIVE FULL Questions, choosing ONE full question from each Module

	PART I	MARKS	OBE	
			CO	RBT
1	Can you overload a constructor? If yes, write a program to illustrate. If no, give reasons. **OR**	[10]	CO1	L1
2	Write a program in Java for a recursive factorial calculator	[10]	CO1	L3
	PART II			
3	Create an interface Bank having a method rateOfInterest() with float return type. Create to classes SBI and PNB which implements interface Bank and return rate of interest of 9.15 and 9.7 respectively. Implement this scenario and print the rate of interest of SBI. **OR**	[10]	CO1	L2
4	Write a program in Java to implement multiple inheritance	[10]	CO1	L2
	PART III			
5	What is a Package? Illustrate with examples how a package can be accessed from outside the package? **OR**	[10]	CO2	L2
6	Discuss Java library package hierarchy and their contents. What are the package member access scopes?	[5+5]	CO1	L2
	PART IV			
7	What is an exception? Explain exception handling mechanism with proper example. **OR**	[10]	CO1	L2
8	Differentiate between throw and throws. Explain with examples.	[10]	CO1	L2
	PART V			
9	Write a Java program to check if a number is Armstrong's number or not? (371 is Armstrong's number as 3*3*3 + 7*7*7 + 1*1*1 = 371) **OR**	[10]	CO2	L1
10	Write a Java program to check if a given string is Pangram. (A string is a pangram string if it contains all the character of the alphabets)	[10]	CO2	L4

1. In Java, we can overload constructors like methods. The constructor overloading can be defined as the concept of having more than one constructor with different parameters so that every constructor can perform a different task.

Example:

```
public class Student {
//instance variables of the class
int id;
String name;

Student(){
System.out.println("this a default constructor");
}

Student(int i, String n){
id = i;
name = n;
}

public static void main(String[] args) {
//object creation
Student s = new Student();
System.out.println("\nDefault Constructor values: \n");
System.out.println("Student Id : "+s.id + "\nStudent Name : "+s.name);

System.out.println("\nParameterized Constructor values: \n");
Student student = new Student(10, "David");
System.out.println("Student Id : "+student.id + "\nStudent Name : "+student.name);
}
}
```

Output:

```
this a default constructor

Default Constructor values:
Student Id : 0
Student Name : null

Parameterized Constructor values:
Student Id : 10
Student Name : David
```

2.

```
class FactorialExample2{
 static int factorial(int n){
  if (n == 0)
    return 1;
  else
    return(n * factorial(n-1));
 }
 public static void main(String args[]){
  int i,fact=1;
  int number=4;//It is the number to calculate factorial
  fact = factorial(number);
```

```
  System.out.println("Factorial of "+number+" is: "+fact);
 }
}
```

3.

```
interface Bank{
        float rateOfInterest();
}
class SBI implements Bank{
        public float rateOfInterest(){return 9.15f;}
}
class PNB implements Bank{
        public float rateOfInterest(){return 9.7f;}
}
class TestInterface2{
        public static void main(String[] args){
                Bank b=new SBI();
                System.out.println("ROI: "+b.rateOfInterest());
        }
}
```

4.

```
interface Printable{
        void print();
}
interface Showable{
        void print();
}
class TestInterface implements Printable, Showable{
        public void print(){System.out.println("Hello");}
        public static void main(String args[]){
                TestInterface obj = new TestInterface();
                obj.print();
        }
}
```

5. Package in Java is a mechanism to encapsulate a group of classes, sub packages and interfaces. Packages are used for:

- Preventing naming conflicts. For example there can be two classes with name Employee in two packages, college.staff.cse.Employee and college.staff.ee.Employee
- Making searching/locating and usage of classes, interfaces, enumerations and annotations easier
- Providing controlled access: protected and default have package level access control. A protected member is accessible by classes in the same package and its subclasses. A default member (without any access specifier) is accessible by classes in the same package only.
- Packages can be considered as data encapsulation (or data-hiding).

There are three ways to access the package from outside the package.

1. import package.*;
2. import package.classname;
3. fully qualified name (static import).

1.

```
//save by A.java
```

```
package pack;
public class A{
  public void msg(){System.out.println("Hello");}

//save by B.java
package mypack;
import pack.*;
class B{
  public static void main(String args[]){
   A obj = new A();
   obj.msg();
  }
}
```

2.

```
//save by A.java
package pack;
public class A{
  public void msg(){System.out.println("Hello");}
}

//save by B.java
package mypack;
import pack.A;
class B{
  public static void main(String args[]){
   A obj = new A();
   obj.msg();
  }
}
```

3.

```
//save by A.java
package pack;
public class A{
  public void msg(){System.out.println("Hello");}
}

//save by B.java
package mypack;
class B{
  public static void main(String args[]){
   pack.A obj = new pack.A();//using fully qualified name
   obj.msg();
  }
}
```

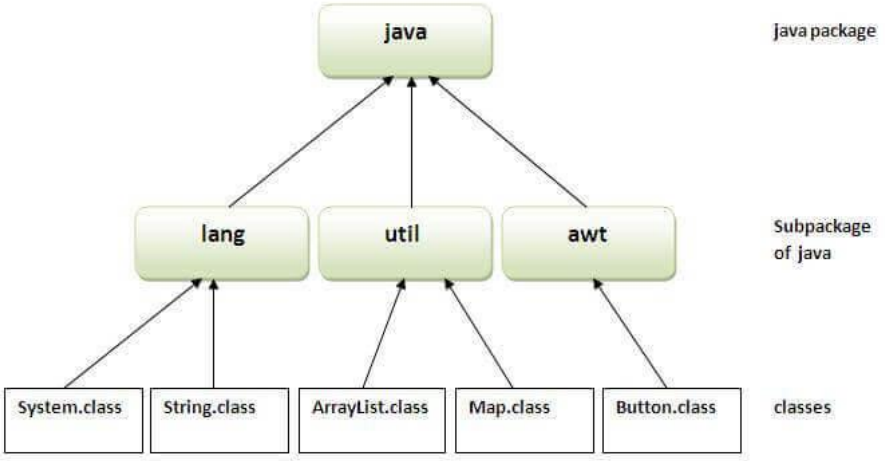
6. Java Built-in Packages:

These packages consist of a large number of classes which are a part of Java API. Some of the commonly used built-in packages are:

1) java.lang: Contains language support classes(e.g classed which defines primitive data types, math operations). This package is automatically imported.

2) java.io: Contains classed for supporting input / output operations.

3) java.util: Contains utility classes which implement data structures like Linked List, Dictionary and support ; for Date / Time operations.
4) java.applet: Contains classes for creating Applets.
5) java.awt: Contain classes for implementing the components for graphical user interfaces (like button , ;menus etc).
6) java.net: Contain classes for supporting networking operations.



Package member access scopes:

	Private Member	Default Member	Protected Member	Public Member
Visible in same class	Yes	Yes	Yes	Yes
Visible in same package in subclass	No	Yes	Yes	Yes
Visible in same package by non-subclass	No	Yes	Yes	Yes
Visible in different package by subclass	No	No	Yes	Yes
Visible in different package by non-subclass	No	No	No	Yes

7. The Exception Handling in Java is one of the powerful mechanism to handle the runtime errors so that the normal flow of the application can be maintained.
An exception is an event that disrupts the normal flow of the program. It is an object which is thrown at runtime.
Exception Handling is a mechanism to handle runtime errors such as ClassNotFoundException, IOException, SQLException, RemoteException, etc.

The core advantage of exception handling is to maintain the normal flow of the application. An exception normally disrupts the normal flow of the application; that is why we need to handle exceptions. Let's consider a scenario:
Suppose there are 10 statements in a Java program and an exception occurs at statement 5; the rest of the code will not be executed, i.e., statements 6 to 10 will not be executed. However, when we perform exception handling, the rest of the statements will be executed. That is why we use exception handling in Java.

```
public class JavaExceptionExample{
  public static void main(String args[]){
   try{
      //code that may raise exception
      int data=100/0;
   }catch(ArithmeticException e){System.out.println(e);}
   //rest code of the program
   System.out.println("rest of the code...");
  }
}
```

8.

Sr. no.	Basis of Differences	throw	throws
1.	Definition	Java throw keyword is used throw an exception explicitly in the code, inside the function or the block of code.	Java throws keyword is used in the method signature to declare an exception which might be thrown by the function while the execution of the code.
2.	Type of exception	Using throw keyword, we can only propagate unchecked exception i.e., the checked exception cannot be propagated using throw only.	Using throws keyword, we can declare both checked and unchecked exceptions. However, the throws keyword can be used to propagate checked exceptions only.
3.	Syntax	The throw keyword is followed by an instance of Exception to be thrown.	The throws keyword is followed by class names of Exceptions to be thrown.
4.	Declaration	throw is used within the method.	throws is used with the method signature.
5.	Internal implementation	We are allowed to throw only one exception at a time i.e. we cannot throw multiple exceptions.	We can declare multiple exceptions using throws keyword that can be thrown by the method. For example, main() throws IOException, SQLException.

Example:

```
public class TestThrowAndThrows
{
   // defining a user-defined method
   // which throws ArithmeticException
   static void method() throws ArithmeticException
   {
      System.out.println("Inside the method()");
      throw new ArithmeticException("throwing ArithmeticException");
   }
public static void main(String args[])
   {
      try
      {
```

```
            method();
        }
        catch(ArithmeticException e)
        {
            System.out.println("caught in main() method");
        }
    }
}
```

9.

```
import java.util.Scanner;
import java.lang.Math;
public class ArmstsrongNumberExample2
{
//function to check if the number is Armstrong or not
static boolean isArmstrong(int n)
{
int temp, digits=0, last=0, sum=0;
//assigning n into a temp variable
temp=n;
//loop execute until the condition becomes false
while(temp>0)
{
temp = temp/10;
digits++;
}
temp = n;
while(temp>0)
{
//determines the last digit from the number
last = temp % 10;
//calculates the power of a number up to digit times and add the resultant to the sum variable
sum +=  (Math.pow(last, digits));
//removes the last digit
temp = temp/10;
}
//compares the sum with n
if(n==sum)
//returns if sum and n are equal
return true;
//returns false if sum and n are not equal
else return false;
}
//driver code
public static void  main(String args[])
{
int num;
Scanner sc= new Scanner(System.in);
System.out.print("Enter the number: ");
//reads the limit from the user
num=sc.nextInt();
if(isArmstrong(num))
{
System.out.print("Armstrong ");
}
```

```
else
{
System.out.print("Not Armstrong ");
}
}
}
```

10.

```
import java.util.Scanner;
public class PangramStringExample4
{
public static void main(String args[])
{
Scanner sc = new Scanner(System.in);
System.out.print("Enter the string: ");
//reads a string from the user
String str = sc.nextLine();
//determines the length of the string
int n=str.length();
int freq[] = new int[26];
//loop iterate over all the characters
for(int i=0;i<26;i++)
{
//initially, each character marked with index 0
freq[i]=0;
}  //end of for loop
//loop iterate over the given string
for(int i=0;i<n;i++)
{
if(str.charAt(i)!=' ')
{
//subtract 'a' from the character at the specified index and increment the freqency index by 1
//means that chracter is marked
freq[str.charAt(i)-'a']++;
}  //end of if
}  //end of for loop
int temp=0;
for(int i=0;i<26;i++)
{
//if frequency index is 0, means character is unmarked
if(freq[i]==0)
{
//set temp to 1 and break the loop
temp=1;
break;
}
}  //end of for loop
//if condition is true, the string is pangram string, else not pangram
if(temp==1)
{
System.out.println("The string is not a pangram string.");
}
else
{
System.out.println("The string is a pangram string.");
```

```
}
}
}
```