BAPATLA ENGINEERING COLLEGE::BAPATLA

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

APRIL-2018- SCHEME OF EVOLUTION

Sub: **OOPS Using JAVA** (14 EC 605) Class: 3/4 ECE

Time: 3Hr Max: 60Marks

1. ONE MARK QUESTIONS AND ANSWERS

a. Difference between the object oriented programming and procedure oriented programming.

Ans)In POP, program is divided into small parts called functions. Importance is not given to data but to functions as well as sequence of actions to be done.

In OOP, program is divided into parts called objects.Importance is given to the data rather than procedures or functions because it works as a real world.

b. Why java is called the "Platform Independent Programming language"?

Ans) Java cannot tied to any hardware or operating system. We can run the java programs on any platform. That's why java is called platform independent programming language.

c. What is for-each loop? Give the syntax.

Ans) for-each is the iteration type of control structure. It is mainly used to traverse array or collection elements. The advantage of for-each loop is that it eliminates the possibility of bugs and makes the code more readable.

Syntax:-

```
int variable;
for(variable initialization ; condition ; increment or decrement)
```

example:

```
int i;
for(i=0;i<5;i++)
```

d. Can I declare the main method of our class as private? Justify.

Ans) Yes, the but in runtime, JVM says main method is not public. So we cannot run it for current application.

e. Explain the purpose of garbage collection in java.

Ans) The garbage collector of JVM discards the unreferenced objects.

f. Define inheritance and give its uses.

Ans) inheritance is the mechanism of deriving a new class from already existing one. The use of inheritance is reusability .

g. Explain nested classes in java.

Ans) The Java programming language allows you to define a class within another class. Such a class is called a nested class.

Example:-

```
classOuterClass {
   ...
classNestedClass {
      ...
   }
}
```

h. State the differences between abstract class and interface in java.

Ans) Variables declared in a Java interface are by default final. An abstract class may contain non-final variables.

Interface can have only abstract methods. Abstract class can have abstract and non-abstract methods

i. Does java support multiple inheritance? justify.

Ans) java does not support multiple inheritance directly due toserious problems to users (dreaded diamond of death). So it is implement by using multiple interfacing.

j. What is the difference between processes and threads?

Ans)

- process is a thread in execution.
- Thread is a set of instruction that has a single flow of control.

k. List the different ways to implement multithreading in java.

Ans)

- By implementing runnable interface
- By extending thread class

l. Explain deadlocks in java.

Ans) sometimes two or more threads need to acquire the locks on several shared objects. This could cause deadlock, in which each thread has the lock on one of the objects and is waiting for the lock on the other object.

UNIT- 1

2. a) Discuss fundamentals principles of oops with respect to java language.

Ans) the fundamental principles of oops are:-

- classes and objects
- data abstraction
- encapsulation
- inheritance
- polymorphism

classes:- class is collection of common features of set of things or objects.

syntax:

```
classclass_name
        {
        Data members;
        Member functions;
         }
```

Objects:- object is the runtime entity that has particular attributes with assigned different values. (Or) object is a instance of a class.

Syntax to create a object:-

```
Return_type  method name(arguments)
                                {
                                Class_name object= new class_name();
                                }
```

Data abstraction:- data abstraction is process of hiding implementation details by showing essential features.

Example: class definition and method declarations

Syntax: Return_type method name(arguments);

Encapsulation:-Binding (or wrapping) code and data together into a single unit is known as encapsulation.

Example: class definitions

Inheritance:-The mechanism of deriving a new class from already existing one is called inheritance. We can derive a new class from existing class using keyword extends.

Syntax:-

```
Class exist
                {
                 }
                Class derive extends exist
                    {
                    }
```

Polymorphism:-Polymorphism is the ability of one task is performed by different ways.

Method overloading and method overriding are examples of polymorphism.

b) Write java program to check the given number is a palindrome or not.

Ans)

```
class Palindrome
{
 public static void main(String args[])
{
  int r,sum=0,temp;
  int n=454;//It is the number variable to be checked for palindrome

  temp=n;
  while(n>0)
{
   r=n%10;  //getting remainder
   sum=(sum*10)+r;
   n=n/10;
  }
  if(temp==sum)
   System.out.println("palindrome number ");
  else
   System.out.println("not palindrome");
}
```

}

(OR)

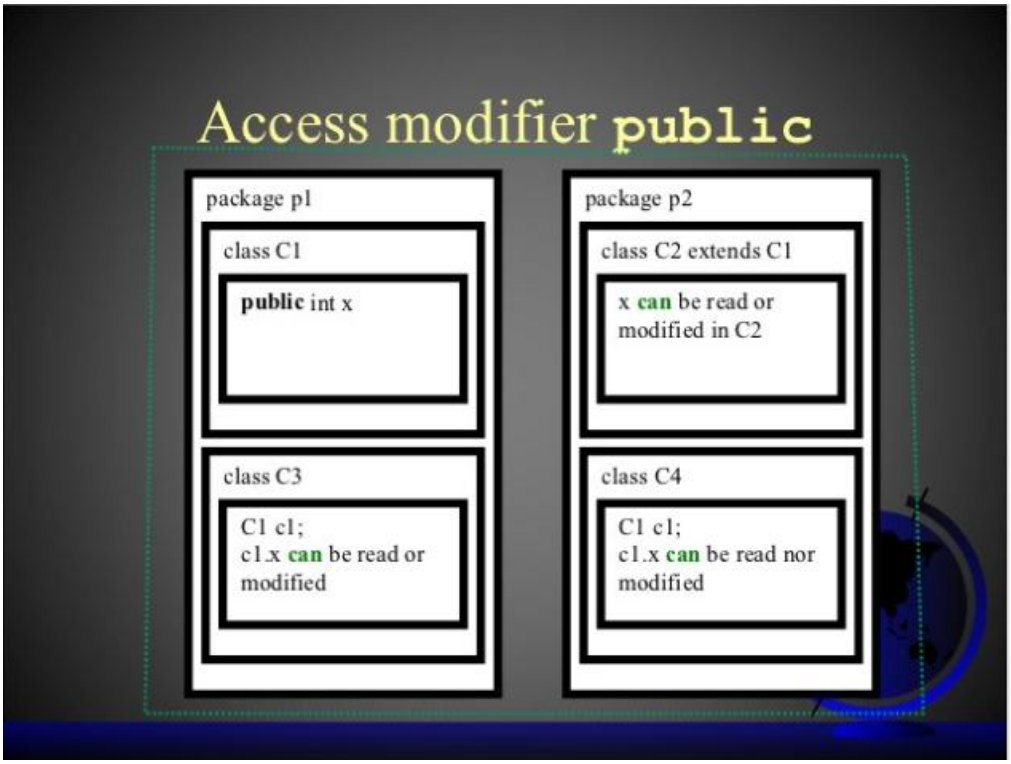
3. a) Explain public, private, protected and default access modifier with an example.
   Ans)The levels of access protection are decides the accessibility of a members in program. The access modifiers affect the accessibility at class and members (methods and instance variables)

   The various levels of access protection available for packages are:-

   - public
   - protected
   - default
   - private

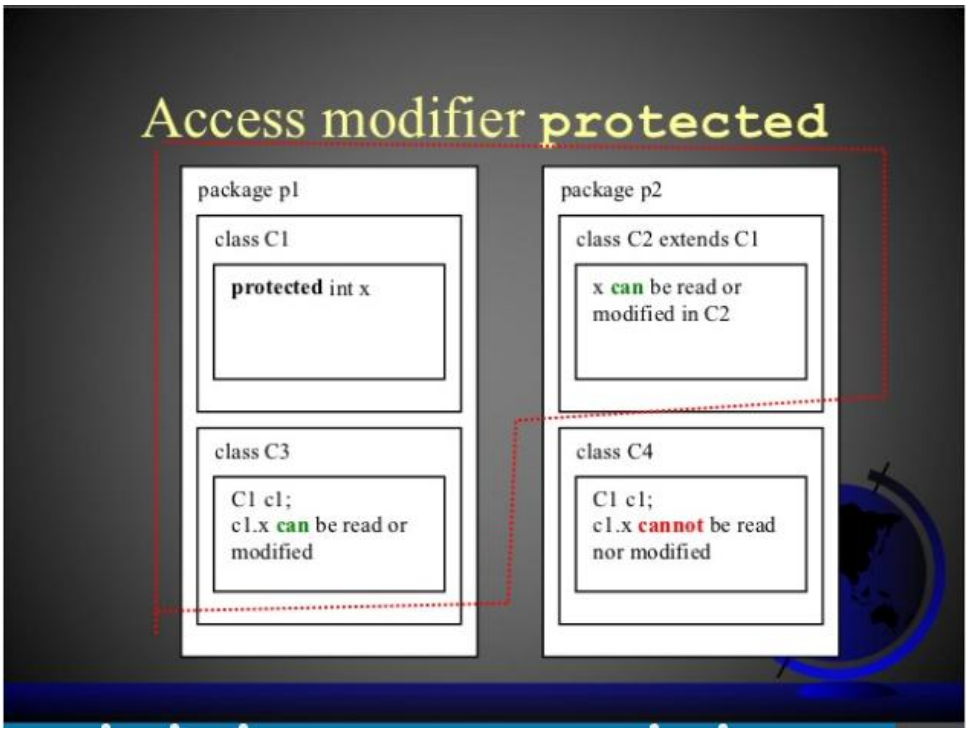
Public:-

- When a class is marked as public, it is visible to all other classes which are inside and outside the package.
- A class marked a public is be accessible to all classes in other packages, but should mention the import statement.
- There is no need for an import statement when a class inside the same package uses the class marked as public



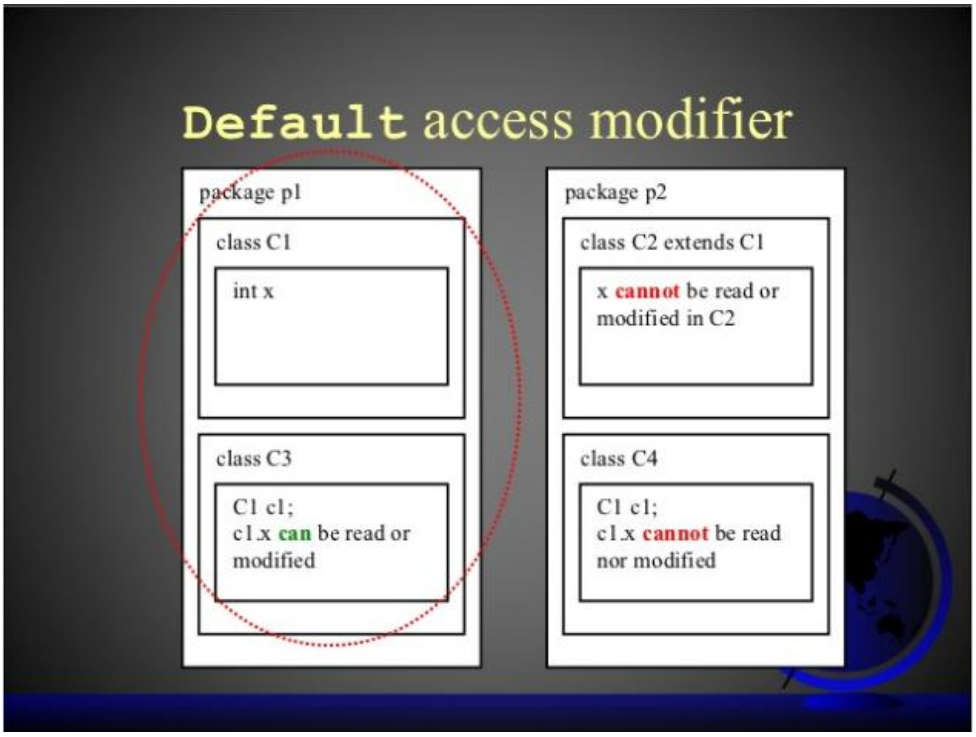
Protected:-

- A member marked as protected is visible to all classes in the same package(just like default).
- Protected members are also accessible to classes outside the package, but the accessing class should be a subclass of the member class.
- The protected members can be accessed only by the subclasses in other packages and can invoke the members only through inheritance mode.



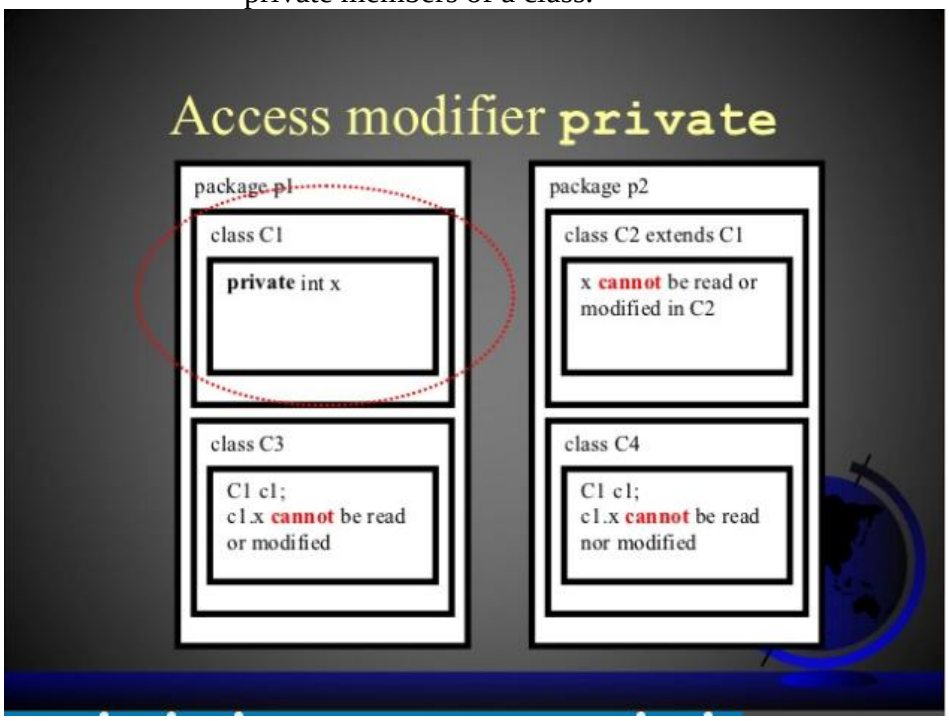
Default:-

- A member marked with default access will be visible to the classes that are in the same package.



Private:-

- When a member is marked as private, it is only visible to other members inside the same class.
- Other classes inside and outside the package will not be able to access the private members of a class.



Access Modifiers - summary

Visibility	public	protected	default	private
From the same class	Yes	Yes	Yes	Yes
From a class in same package	Yes	Yes	Yes	No
From a class from outside the package	Yes	No	No	No
From a subclass outside the package	Yes	Yes(Only through inheritance mode)	No	No
From a subclass in the same package	Yes	Yes	Yes	No

b) Explain control statements in java with suitable examples.

Ans) Control Statements

The control statement are used to controll the flow of execution of the program . This execution order depends on the supplied data values and the conditional logic. Java contains the following types of control statements:

1- Selection Statements
2- Repetition Statements
3- Branching Statements

Selection statements:

If Statement:

This is a control statement to execute a single statement or a block of code, when the given condition is true and if it is false then it skips if block and rest code of program is executed .

Syntax:

```
if(conditional_expression)
{
<statements>;
  ...;
  ...;
}
```

Example: If n%2 evaluates to 0 then the "if" block is executed. Here it evaluates to 0 so if block is executed. Hence "This is even number" is printed on the screen.

```
int n = 10;
if(n%2 = = 0){

System.out.println("This is even number");

}
```

If-else Statement:

The "if-else" statement is an extension of if statement that provides another option when 'if' statement evaluates to "false" i.e. else block is executed if "if" statement is false.

Syntax:

```
if(conditional_expression){
<statements>;
  ...;
  ...;
  }
else{
<statements>;
  ....;
  ....;
   }
```

Example: If n%2 doesn't evaluate to 0 then else block is executed. Here n%2 evaluates to 1 that is not equal to 0 so else block is executed. So "This is not even number" is printed on the screen.

```
int n = 11;
if(n%2 = = 0){

System.out.println("This is even number");

}

else{

System.out.println("This is not even number");

}
```

Switch Statement:

This is an easier implementation to the if-else statements. The keyword "switch" is followed by an expression that should evaluates to byte, short, char or int primitive

data types ,only. In a switch block there can be one or more labeled cases. The expression that creates labels for the case must be unique. The switch expression is matched with each case label. Only the matched case is executed ,if no case matches then the default statement (if present) is executed.

Syntax:

```
switch(control_expression){
case expression 1:
<statement>;
case expression 2:
<statement>;
   ...
   ...
case expression n:
<statement>;
default:
<statement>;
  }//end switch
```

Example: Here expression "day" in switch statement evaluates to 5 which matches with a case labeled "5" so code in case 5 is executed that results to output "Friday" on the screen.

```
int day = 5;
switch (day) {
case 1:
System.out.println("Monday");
break;
case 2:
System.out.println("Tuesday");
break;
case 3:
System.out.println("Wednesday");
break;
case 4:
System.out.println("Thrusday");
break;
case 5:
System.out.println("Friday");
break;
case 6:
System.out.println("Saturday");
break;
case 7:
System.out.println("Sunday");
break;
default:
System.out.println("Invalid entry");
break;
}
```

Repetition Statements:

while loop statements:
This is a looping or repeating statement. It executes a block of code or statements till the given condition is true. The expression must be evaluated to a boolean value. It continues testing the condition and executes the block of code. When the expression results to false control comes out of loop.

Syntax:
```
while(expression){
<statement>;
  ...;
  ...;
  }
```

Example: Here expression i<=10 is the condition which is checked before entering into the loop statements. When i is greater than value 10 control comes out of loop and next statement is executed. So here i contains value "1" which is less than number "10" so control goes inside of the loop and prints current value of i and increments value of i. Now again control comes back to the loop and condition is checked. This procedure continues until i becomes greater than value "10". So this loop prints values 1 to 10 on the screen.

```
int i = 1;
//print 1 to 10

while (i <= 10){

System.out.println("Num " + i);

i++;

}
```

do-while loop statements:
This is another looping statement that tests the given condition past so you can say that the do-while looping statement is a past-test loop statement. First the do block statements are executed then the condition given in while statement is checked. So in this case, even the condition is false in the first attempt, do block of code is executed at least once.

Syntax:
```
do{
<statement>;
  ...;
  ...;
}while (expression);
```

Example: Here first do block of code is executed and current value "1" is printed then the condition i<=10 is checked. Here "1" is less than number "10" so the control comes back to do block. This process continues till value of i becomes greater than 10.

```
int i = 1;
do{

System.out.println("Num: " + i);

i++;

}while(i <= 10);
```

for loop statements:
This is also a loop statement that provides a compact way to iterate over a range of values. From a user point of view, this is reliable because it executes the statements within this block repeatedly till the specified conditions is true .

Syntax:

```
for (initialization; condition; increment or decrement){
<statement>;
  ...;
  ...;
  }
```

initialization: The loop is started with the value specified.
condition: It evaluates to either 'true' or 'false'. If it is false then the loop is terminated.
increment or decrement: After each iteration, value increments or decrements.

Example: Here num is initialized to value "1", condition is checked whether num<=10. If it is so then control goes into the loop and current value of num is printed. Now num is incremented and checked again whether num<=10.If it is so then again it enters into the loop. This process continues till num>10. It prints values 1 to10 on the screen.

```
for (intnum = 1; num<= 10; num++){
System.out.println("Num: " + num);

}
```

Branching Statements:

Break statements:
The break statement is a branching statement that contains two forms: labeled and unlabeled. The break statement is used for breaking the execution of a loop (while, do-while and for) . It also terminates the switch statements.

Syntax:

```
break;  // breaks the innermost loop or switch statement.
```

break label; // breaks the outermost loop in a series of nested loops.

Example: When if statement evaluates to true it prints "data is found" and comes out of the loop and executes the statements just following the loop.

Continue statements:
This is a branching statement that are used in the looping statements (while, do-while and for) to skip the current iteration of the loop and resume the next iteration .

Syntax:
continue;

Example:

Return statements:
It is a special branching statement that transfers the control to the caller of the method. This statement is used to return a value to the caller method and terminates execution of method. This has two forms: one that returns a value and the other that can not return. the returned value type must match the return type of method.

Syntax:
return;
return values;

return; //This returns nothing. So this can be used when method is declared with void return type.
return expression; //It returns the value evaluated from the expression.

Example: Here Welcome() function is called within println() function which returns a String value "Welcome to roseIndia.net". This is printed to the screen.

UNIT-2

4. a) Discuss the use of this keyword with an example.
   Ans) This can be used to refer current class instance variable. This can be used to invoke current class method (implicitly). Generally we use this keyword when the instance variables and local arguments of a member function are declared as same. This is illustrated in below example program.
   Example:
   packagethiscomplex;

   class complex
   {
   intreal,img;
   void read(intreal,intimg)

```
    {
this.real=real;
this.img=img;
    }

complex sum(complex c1,complex c2)
    {
this.real=c1.real+c2.real;
this.img=c1.img+c2.img;
return this;

    }
void display()
    {
System.out.println(real+"+i"+img);
    }

}

public class Thiscomplex
{

public static void main(String[] args)
    {
complex k=new complex();

complex l=new complex();

k.read(5,6);
l.read(3,4);
complex add=new complex();
add=add.sum(k,l);
k.display();
l.display();
add.display();
    }

}
```

b) Explain constructor overloading with suitable example.

Ans)

Constructor is a special member function by which initialize the objets itself. Constructors are invoked automatically whenever the objects are created.

Rules for defining constructors:

1. name the constructor as same as class name
2. there is no need to mention return type to constructor
3. constructors are automatically invokes whenever the objects are declared.

Constructor overloading:
Generally method overloading means defining different methods with same method name and different arguments. Here constructor name is class name. so if you define the two or more constructors in same class with different arguments then constructors will be automatically overloaded.
Example:-

```
package constructor;

public class Constructor
{



public static void main(String[] args)
   {
     Student s=new Student();
      Student s1=new Student(111,"Raj");
      Student s2=new Student(222,"Arya",25);
     Student s3=new Student(s1);
s1.display();
s2.display();
s3.display();

   }
}
class Student
{
int id;
   String name;
int age;
Student(inti,String n)
   {
id=i;
name=n;
   }
Student(inti,Stringn,int a)
   {
id=i;
name=n;
age=a;
   }
Student()
   {
System.out.println("its a default costructor");
   }
Student(Student c)
   {
id=c.id;
name=c.name;
   }
```

```
void display()
   {
System.out.println(id+" "+name+" "+age);
   }

}
```

(OR)

5. a) .What is general form of a class? How objects are declared explain with an example.

Ans) class: class is a collection of common properties of set of objects.

General form:

```
classclass_name
{
Data members;
Member functions;
}
```

Data members:- the variables which are declared inside the class are called datamembers.

Member functions:- the functions which are declared or defined inside a class are called member functions.

Object:-

The data members and member functions of a class can be accessed at outside of class through the **instance of a class**.

The instance of a class is called **Object.** Object is a runtime entity which is having particular properties that may be assigned different values.

Syntax:

```
Class_name object=new class_name();
```

Example:

```
package object;
importjava.util.*;

class Object
{

public static void main(String[] args)
   {
int i;
      Scanner sc=new Scanner(System.in);

doublevol;
   Box []mybox=new Box[60];

for(i=0;i<3;i++)
   {
mybox[i]=new Box();
System.out.println("enter the dimensions of "+i+"th Box:");
```

```
mybox[i].width=sc.nextInt();
mybox[i].height=sc.nextInt();
mybox[i].depth=sc.nextInt();
   }
for(i=0;i<3;i++)
   {
vol=mybox[i].volume();
System.out.println("the volume of "+i+"th box is:"+vol);
   }

   }

}
class Box
{

double width;
double height;
double depth;
double volume()
   {
return width*height*depth;
   }
}
```

b) Write a java program to swap three numbers without using fourth variable.
Ans)

```
public class JavaSwapExample {

public static void main(String[] args) {

int x = 20;
int y = 10;
int z=30;

System.out.println("Before Swapping");
System.out.println("Value of x is :" + x);
System.out.println("Value of y is :" + y);
System.out.println("Value of y is :" + z);

  //add both the numbers and assign it to first
  x = x + y+z;
  y = x – y– z;
z = x – y– z;
z = x – y– z;
System.out.println("Before Swapping");
```

```
System.out.println("Value of x is :" + x);
System.out.println("Value of y is :" + y);
System.out.println("Value of y is :" + z);

 }
}
```

UNIT-3

6. a) Explain usage of super keyword with suitable example.

Ans)

Super:- super is a keyword which is used to refer the parent class. By using the super keyword we can access the overridden data of super class in child class.

Example:-

```
packagesuperkeyword;
class base
{
int a;
base()
   {
      a=0;
System.out.println("the a value is:"+a);
   }

}
class derived extends base
{
int b;
derived()
   {
super();
      b=1;
System.out.println("the b value is:"+b);
   }

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

derived d=new derived();

   }

}
```

b) Write a java program to find the area and perimeter of square and circle using interface.

Ans)

```
interface area
     {
double pi = 3.14;
doublecalc(double x,double y);
doublecalcsqr(double x);
      }
interface perimeter
{
double pi = 3.14;
doublegetperimeter(double pi,double radius);
public double getperimeter(double length);
}

classsqr implements area,perimeter
  {
public double calcsqr(double x)
       {
return(x*x);
        }
public double getperimeter(double length)
{
return(4*length);
  }

classcir implements area ,perimeter
  {
public double calc(double x,double y)
       {
return(pi*x*x);
        }
public double getperimeter(double pi,double radius)
{
return(2*pi*radius);
}
  }

class test7
 {
public static void main(String arg[])
      {
sqr r = new sqr();
cir c = new cir();
area a;

         a = r;
System.out.println("\nArea of square is : " +a.calcsqr(10));
System.out.println("\nperimeter of square is : " +a.getperimeter(15));
         a = c;
System.out.println("\nArea of Circle is : " +a.calc(15,15));
```

```
System.out.println("\nperimeter of Circle is : " +a.getperimeter(3.14,15));
      }
  }
```

(OR)

7. a) Explain creating packages and accessing packages with example.
   Ans)A java package is a group of similar types of classes, interfaces. Package in java can be categorized in two form, built-in package and user-defined package.

   The package keyword is used to create a package in java.

   Syntax to create a package is
   packagepackage_name;
   class definitions

   example:-

```
packagemypack;
public class Simple
{
public static void main(String args[])
{
System.out.println("Welcome to package");
  }
}
```

   Accessing packages:-
   We can access the packages in two ways
   a. importing entire package
   b. importing particular class in package

   - if you want to import all classes in package we need to entire package like below

   importpackage_name.*;

   - if you want to import only particular class in package we import like below
     importpackage_name.class_name;

example:-

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

class B{
public static void main(String args[]){
  A obj = new A();
```

```
obj.msg();
  }
}
```

b) Define interfaces in java. How interfaces are implemented? How they can be accessed?

Ans)

Interfaces are similar to abstract classes, but differ in their functionality. In interfaces, none of the methods are implemented means interfaces declares methods without body.

Interfaces are syntactically similar to classes but they lack instance variables, and their methods are declared without any body. But it can contain final variables, which must be initialized with values.

Once it defined any number of classes can implement an interface. One class can implement any number of interfaces. If we are implementing an interface in a class we must implement all the methods defined in the interface as well as a class can also implement its own methods.

Syntax to define interface :-

```
interfaceinterface_name
{
Final variables;
Method declarations;
}
```

Syntax to implement interface;-

```
Class A implements interface_name
{
Instance variables;
Definitions of interface methods;
Definitions of own methods;
}
```

Example:-

```
package hybrid;


public class Hybrid
{

public static void main(String[] args)
   {

      Results student1=new Results();
student1.getNumber(1234);
student1.getMarks(27.5F,33.0F);
student1.display();
   }
```

```
}
class Student
{
introllNumber;
voidgetNumber(int n)
   {
rollNumber=n;
   }
voidputNumber()
   {
System.out.println("roll no is "+rollNumber);
   }
}
class Test extends Student
{
float part1,part2;
voidgetMarks(float m1,float m2)
   {
      part1=m1;
      part2=m2;
   }
voidputMarks()
   {
System.out.println("marks obtained");
System.out.println("part1=  "+part1);
System.out.println("part2=  "+part2);
   }
}
interface Sports
{
final float sportwt=6.0F;
voidputwt();
}
class Results extends Test implements Sports
{
float total;
public void putwt()
   {
System.out.println("Sports wt =  "+sportwt);
   }
void display()
   {
total=part1+part2+sportwt;
putNumber();
putMarks();
putwt();
System.out.println("total score = "+total);
   }
}
```

UNIT-4

8. a) What is an exception? Explain the usage of try catch block in exception handling with help of java program.

Ans)exception is a runtime error which arises during execution of java program. The term exception in java stands for an “exceptional event”. So exceptions are nothing but some abnormal and typically an event or conditions that arise during the execution which may interrupt the normal flow of program.

Exception turns the direction of normal flow of the program control and send to the related catch() block and should display error message for taking proper action. This process is known as Exception handling.

Try:- piece of code of your program that you want to monitor for exceptions are contained within a try block. If an exception occurs within the try block, it is thrown.
Catch:- catch block can catch this exception and handle it in logical manner

Example:-

```
class ExceptionDemo1
        {
        public static void main(String args[])
        {
        try{
        int num1=30, num2=0;
        int output=num1/num2;
        System.out.println ("Result = " +output);
         }


        catch(ArithmeticException e){
        System.out.println ("Arithmetic Exception: You can't divide an integer by 0");
          }
        try{
        int a[]=new int[10];
          //Array has only 10 elements
        a[11] = 9;
         }

        catch(ArrayIndexOutOfBoundsException e){
        System.out.println ("ArrayIndexOutOfBounds");
        }
        try{
        intnum=Integer.parseInt ("XYZ") ;
        System.out.println(num);
         }
        catch(NumberFormatException e){
        System.out.println("Number format exception occurred");
        }
        try{
         String str="easysteps2buildwebsite";
```

```
            System.out.println(str.length());;
            char c = str.charAt(0);
             c = str.charAt(40);
            System.out.println(c);
            }
            catch(StringIndexOutOfBoundsException e){
            System.out.println("StringIndexOutOfBoundsException!!");
              }
            try{
                    String str=null;
                    System.out.println (str.length());
            }catch(NullPointerException e){
                    System.out.println("NullPointerException..");
            }

        }
    }
```

b) what do understand by synchronization? Discuss with an example.

Ans)

Synchronization in java is the capability of control the access of multiple threads to any shared resource. It is better option where we want to allow only one thread to access the shared resource.

We use this synchronization to prevent thread interference and to prevent consistency problem.

```
class Table{
 synchronized void printTable(int n){//synchronized method
   for(int i=1;i<=5;i++){
     System.out.println(n*i);
     try{
      Thread.sleep(400);
     }catch(Exception e){System.out.println(e);}
   }

 }
}
```

```
class MyThread1 extends Thread{
Table t;
     MyThread1(Table t){
     this.t=t;
     }
     public void run(){
     t.printTable(2);
     }
}
class MyThread2 extends Thread{
Table t;
     MyThread2(Table t){
     this.t=t;
     }
     public void run(){
     t.printTable(100);
     }
}

public class TestSynchronization{
public static void main(String args[]){
    Table obj = new Table();//only one object
    MyThread1 t1=new MyThread1(obj);
    MyThread2 t2=new MyThread2(obj);
    t1.start();
    t2.start();
    }
}
```

(OR)

9. a) Explain thread life cycle methods.

Ans) A thread can be in one of the five states. According to sun, there is only 4 states in thread life cycle in java new, runnable, non-runnable and terminated. There is no running state.

But for better understanding the threads, we are explaining it in the 5 states.

The life cycle of the thread in java is controlled by JVM. The java thread states are as follows:

New

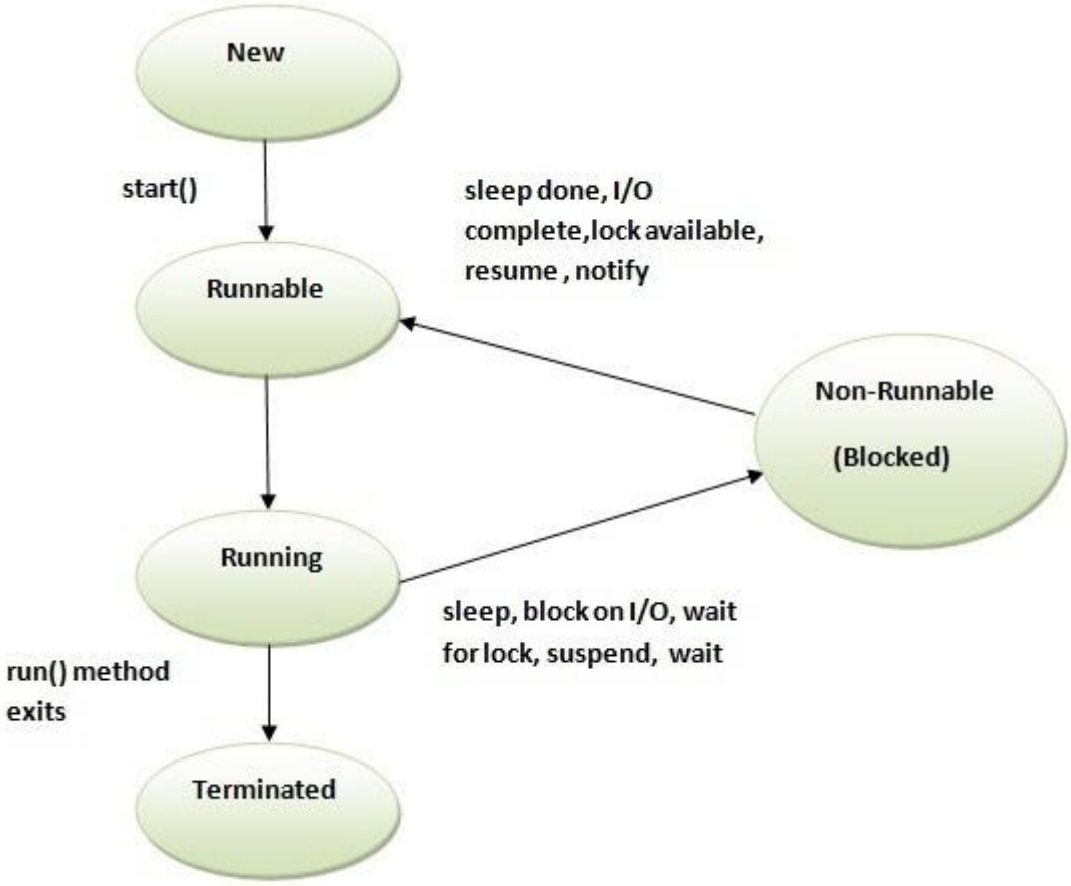
Runnable

Running

Non-Runnable (Blocked)

Terminated

thread life cycle in java:



1) New

The thread is in new state if you create an instance of Thread class but before the invocation of start() method.

2) Runnable

The thread is in runnable state after invocation of start() method, but the thread scheduler has not selected it to be the running thread.

3) Running

The thread is in running state if the thread scheduler has selected it.

4) Non-Runnable (Blocked)

This is the state when the thread is still alive, but is currently not eligible to run.

5) Terminated

A thread is in terminated or dead state when its run() method exits.

b) write java program to implement producer consumer problem using threads.

Ans)
```
public class ProducerConsumerTest {
public static void main(String[] args) {
CubbyHole c = new CubbyHole();
     Producer p1 = new Producer(c, 1);
     Consumer c1 = new Consumer(c, 1);
p1.start();
c1.start();
  }
}
classCubbyHole {
privateint contents;
privateboolean available = false;

public synchronized int get() {
while (available == false) {
try {
wait();
       } catch (InterruptedException e) {}
```

```
      }
available = false;
notifyAll();
return contents;
   }
public synchronized void put(int value) {
while (available == true) {
try {
wait();
         } catch (InterruptedException e) { }
      }
contents = value;
available = true;
notifyAll();
   }
}
class Consumer extends Thread {
privateCubbyHole cubbyhole;
privateint number;

public Consumer(CubbyHole c, int number) {
cubbyhole = c;
this.number = number;
   }
public void run() {
int value = 0;
for (int i = 0; i < 10; i++) {
value = cubbyhole.get();
System.out.println("Consumer #" + this.number + " got: " + value);
      }
   }
}
class Producer extends Thread {
privateCubbyHole cubbyhole;
privateint number;
public Producer(CubbyHole c, int number) {
cubbyhole = c;
this.number = number;
   }
public void run() {
for (int i = 0; i < 10; i++) {
cubbyhole.put(i);
System.out.println("Producer #" + this.number + " put: " + i);
try {
sleep((int)(Math.random() * 100));
         } catch (InterruptedException e) { }
      }
   }
}
```