import java.util.Collections;

import java.util.Comparator;

import java.util.LinkedList;

import java.util.List;

import java.util.ArrayList;

import java.util.Vector;

public class Books implements Comparable<Books> {

private String title;//local variable

private String author;

private double edition;

private int yearPublication;

public Books(long isbn, String title, String author, int yearPublication, double price){

this.title = title;

this.author = author;

this.price = edition;

this.yearPublication = yearPublication;

}

public String getTitle(){

return title;

}

public String getAuthor(){

return author;

}

public double getEdition(){

return edition;

}

public int getYearPublication(){

return yearPublication;

}

public String toString(){

return "\n\tTitle: " + title + "\n\tAuthor: " + author + "\n\tEdition: " + edition + "\n\tYear of publication : " + yearPublication ;

}

public int compareTo(Books B)/>

{

return getTitle().compareTo(((Books)B)/>.getTitle());

}

public static Comparator<Books>CompareYrPublished=new Comparator<Books>()

{

public int compare(Books b1,Books b2)

{

return b1.getYearPublication()-b2.getYearPublication();

}

};

public static void main(String[] args)

{

List <Books> book=new LinkedList<Books>(); //Declaration of book linkedlist inside Books linked list.

Books b1=new Books(1456,"cooking Tips","James",2010,15);

Books b2=new Books(14416,"Adventure","Nelly",2012,112);

Books b3=new Books(1444,"Bye Forever","John",2011,12);

book.add(b1);

book.add(b2);

book.add(b3);

Comparator CompareYrPublished = Collections.reverseOrder();

Collections.sort(book,Books.CompareYrPublished);

for(int i=0;i<book.size();i++)

{

System.out.println(book.get(i));

}

}

} // end class Book

**Polymorphism means an object having many different types. This concept can be used for code reusability and hiding implementation details. Given that there is a ShapeUtilty class with a method called process(), a Triangle class, a Circle class, a Rectangle class, and an interface Processable - with a method called findArea(). Use ShapeUtility's process method to calculate the area of any shapes.**

**Answer:**

**Main Class:**

public class ShapeUtility {

public static void main(String[] args){

Triangle myTriangle =new Triangle();

Circle myCircle = new Circle();

Rectangle myRectangle=new Rectangle();

public void Process(){

myTriangle.findArea(10,5);

myCircle.findArea(5);

myRectangle.findArea(10,5);

}

}

}

**Interface:**

public interface Processable{

public int findArea(int param);

}

**Triangle Class:**

public class Triangle implements Processable{

int base;//instance variable

int height;

public int findArea(int a, int b){

base= a;

height=b;

int c= (1/2 \* base \*height);

System.out.println(“Area is:” +c);

}

}

**Circle Class:**

public class Circle implements Processable{

int radius;//instance variable

public int findArea(int a){

radius= a;

int c= (3.1416\*radius\*radius);

System.out.println(“Area is:” +c);

}

}

**Rectangle Class:**

public class Rectangle implements Processable{

int width;//instance variable

int height;

public int findArea(int a, int b){

width= a;

height=b;

int c= (width \*height);

System.out.println(“Area is:” +c);

}

}