Practice Problems: Inheritance & Polymorphism

|  |  |
| --- | --- |
| public class Foo {  public void method1() {  System.out.println("foo 1");  }  public void method2() {  System.out.println("foo 2");  }  public String toString() {  return "foo";  }  }  public class Bar extends Foo {  public void method2() {  System.out.println("bar 2");  }  }  public class Baz extends Foo {  public void method1() {  System.out.println("baz 1");  }  public String toString() {  return "baz";  }  }  public class Mumble extends Baz {  public void method2() {  System.out.println("mumble 2");  }  }  public class Polymorphism{  public static void main(String [] args){  Foo[] pity = { new Baz(), new Bar(),  new Mumble(), new Foo() };  for (int i = 0; i < pity.length; i++) {  System.out.println(pity[i]);  pity[i].method1();  pity[i].method2();  System.out.println();  }  } | baz  baz 1  foo 2  foo  foo 1  bar 2  baz  baz 1  mumble 2  foo  foo 1  foo 2  \*( ) method is inherited. Otherwise, method is overridden. |

# Tracing programs: The above is the program demonstrated in class. Now, what gets printed to the screen when we execute the following classes on the left?

|  |  |
| --- | --- |
| public class A { public int x = 1; public void setX(int a){ x=a; } }  public class B extends A { public int getB(){ setX(2); return x;} }  public class C { //trace program 1 public static void main(String [] args){ A a = new A(); B b = new B(); System.out.println(a.x); System.out.println(b.getB()); } } | 1  2 |
| public class A { private int x = 1; protected void setX(int a){ x=a; } protected int getX(){ return x;} }  public class B extends A { public int getB(){ setX(2); return getX(); } }  public class C { //trace program 2 public static void main(String [] args){ A a = new A(); B b = new B(); System.out.println(a.getX()); System.out.println(b.getB()); } } | 1  2 |
| public class A { protected int x = 1;  protected void setX(int a){x=a;} protected int getX(){return x;} }  public class B extends A { public int getB(){ setX(2); return x;  } }  public class C { //trace program 3 public static void main(String [] args){ A a = new A(); B b = new B(); System.out.println(a.getX()); System.out.println(b.x);  System.out.println(b.getB()); } } | 1  1  2 |
| public class A { protected int x = 1; protected void setX(int a){ x=a; } protected int getX(){ return x;} }  public class B extends A { protected int x = 3; public int getX(){ return x; } public int getB(){ return x; } }  public class C { //trace program 4 public static void main(String [] args){ A a = new A(); B b = new B(); System.out.println(a.getX()); System.out.println(b.getB()); System.out.println(b.getX());  System.out.println(a.x);  System.out.println(b.x);  } } | 1  3  3  1  3 |
| public class A { protected int x = 1; protected void setX(int a){ x=a; } protected int getX(){ return x;} }  public class B extends A { protected int x = 3; public int getX(){ return x; } public int getB(){ return x; } }  public class C { //trace program 5 public static void main(String [] args){ A a = new A(); A b = new B();  System.out.println(a.getX()); System.out.println(b.getX()); System.out.println(a.x); System.out.println(b.x);  } } | 1  3  1  1 |
|  |  |

|  |  |
| --- | --- |
| public class A { protected int x = 1; protected void setX(int a){ x=a; } protected int getX(){ return x;} }  public class B extends A { protected int x = 3; public int getX(){ setX(2);  return x; }  public int getB(){ return x; } }  public class C { //trace program 6 public static void main(String [] args){ A a = new A(); A b = new B();  System.out.println(a.getX()); System.out.println(b.getX()); System.out.println(a.x); System.out.println(b.x);  } } | 1  3  1  2 |
| public class Ham {  public void a() {  System.out.println("Ham a");  }  public void b() {  System.out.println("Ham b");  }  public String toString() {  return "Ham";  }  }  public class Lamb extends Ham {  public void b() {  System.out.println("Lamb b");  }  }  public class Yam extends Lamb {  public void a() {  System.out.println("Yam a");  }  public String toString() {  return "Yam";  }  }  public class Spam extends Yam {  public void a() {  System.out.println("Spam a");  }  }  public class Polymorphism2 {  //trace program 7  public static void main (String [] args){  Ham[] food = { new Spam(), new Yam(),  new Ham(), new Lamb() };  for (int i = 0; i < food.length; i++) {  System.out.println(food[i]);  food[i].a();  food[i].b();  System.out.println();  }  }  } |  |

|  |  |
| --- | --- |
| public class Ham {  int a = 0;  int b = 1;  public void a() {  System.out.println("Ham “ + a);  }  public void b() {  System.out.println("Ham “ + b);  }  public String toString() {  return "Ham “ + a + “ “ + b;  }  }  public class Spam extends Ham {  int a = 2;  public void a() {  System.out.println("Spam “ +a);  }  }  public class Yam extends Spam {  int b = 3;  public void a() {  System.out.println("Yam “ + a);  }  public void b() {  System.out.println(“Yam “ + b);  }  }  public class Polymorphism3 {  //trace program 8  public static void main (String [] args){  Ham[] food = { new Spam(), new Yam(),  new Ham()};  for (int i = 0; i < food.length; i++) {  System.out.println(food[i]);  food[i].a();  food[i].b();  System.out.println(food[i].a);  System.out.println(food[i].b);  System.out.println();  }  }  } | Ham 01  Spam 2  Ham1  0  1  Ham 01  Yam 2  Yam3  0  1  Ham 01  Ham 0  Ham 1  0  1 |

|  |  |
| --- | --- |
| public class A  {  private String x = "Ax";  protected String y = "Ay";  public String z = "Az";  public String toString() {  return x + y + z;  }  public static void main(  String [] args)  {  A a = new A();  System.out.println(a);  }  }  public class B extends A  {  private String x = "Bx";  public String z = "Bz";  public String toString() {  return x + y + z;  }  public static void main(  String [] args)  {  B b = new B();  System.out.println(b);  }  } | public class C extends A  {  private String x = "Cx";  public static void main(  String [] args)  {  C c = new C();  System.out.println(c.x);  System.out.println(c);  }  }  public class D extends C  {  //trace program 9  private String x = "Dx";  public String z = "Dz";  public static void main(  String [] args)  {  D d = new D();  System.out.println(d.x);  System.out.println(d.y);  System.out.println(d.z);  System.out.println(d);  C c = new D();  // Error: System.out.println(c.x);  System.out.println(c.y);  System.out.println(c.z);  System.out.println(c);  }  } |

# Program Development

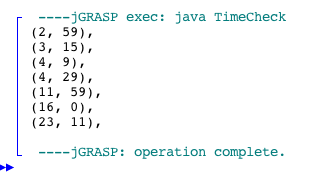
1. Below is a partially filled-out Time class, which stores the hour and minute in private attributes. The constructor is also given.

public class Time implements Comparable {  
private int hour =0;  
private int minute =0;  
public Time (int h, int m){  
 if (h<0||h>23) return;  
 if(m<0||m>59) return;  
 hour = h;  
 minute = m;  
}  
// additional (instance) methods will go here.  
…

}

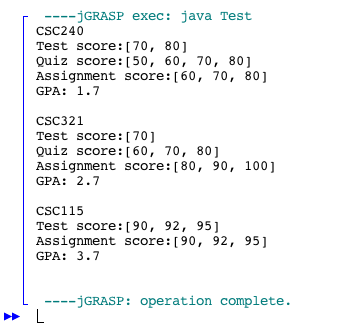
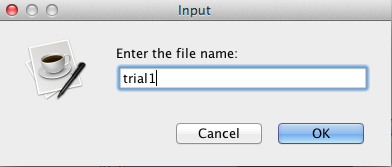
Please complete the above Time class with methods compareTo and toString, in order to support the following application class and reach the desired result:

import java.util.Arrays;  
  
public class TimeCheck{  
public static void main(String args[]){  
 Time [] t = {new Time(3, 15), new Time(16, 0), new Time(4, 29),  
 new Time(11, 59), new Time(23, 11), new Time(2, 59), new Time (4, 9) };  
  
 Arrays.sort(t);  
 for (Time f: t)  
 System.out.println(f+",");   
}  
}



1. Given the super class Student (contained in the package that is available at [www.cs.wcupa.edu/~zjiang/240proj3.zip](http://www.cs.wcupa.edu/~zjiang/240proj3.zip)). Please complete the following class Records, in order to support the weighted score grading by the Test program and reach the desire result by using the input file “trial1”.

import javax.swing.JOptionPane;  
import java.io.\*;  
import java.util.\*;  
  
public class Test{  
public static void main(String [] args){  
Records [] t = {  
 new Records("CSC240", 2, 4, 3, 60, 10, 30),   
 // 2 tests, 4 quizzes, and 3 assignments  
 new Records("CSC321", 1, 3, 3, 25, 15, 60),   
 // 1 test, 3 quizzes, and 3 assignments  
 new Records("CSC115", 3, 0, 3, 60, 0, 40)};   
 // 3 tests, 0 quiz, and 3 assignments  
  
 String filename =   
 JOptionPane.showInputDialog("Enter the file name: ");  
 File inputFile = new File (filename);  
 try {   
 Scanner input = new Scanner(inputFile);  
 String val;  
 while(input.hasNext()){  
 val = input.nextLine();  
 String [] data = val.split(" ");  
   
 int index = Integer.parseInt(data[0]);  
 char opt = data[1].toLowerCase().charAt(0);  
 int [] s = new int[data.length-2];  
 for(int i = 0; i<s.length; i++)  
 s[i] = Integer.parseInt(data[i+2]);  
   
 switch(opt){  
 case 't':  
 t[index].set\_test(s);  
 break;  
 case 'q':  
 t[index].set\_quiz(s);  
 break;  
 case 'a':  
 t[index].set\_assignment(s);  
 break;  
 }  
 }  
 input.close();  
 }  
 catch (FileNotFoundException e)  
 {  
 System.out.println("file reading fails.");  
 }   
 for(int i = 0; i<t.length; i++){  
 Records tmp = t[i];  
 tmp.grading();  
 System.out.println(tmp);  
 }  
}  
}



In detail, the class Records should have the private attributes and the constructor as follows:

import java.util.Arrays;  
  
public class Records extends Student{  
private int [] test = null;  
private int [] quiz = null;  
private int [] assignment = null;  
  
public Records (String name, int tn, int qn, int an, int tw, int qw, int aw){  
 super(name, tw, qw, aw);  
 test = new int [tn];  
 quiz = new int [qn];  
 assignment = new int [an];  
}  
…

}

This class should also have the following methods:

* 1. set\_test: that will accept a group of test scores in an integer array. The values will copy to the private attribute array test.
  2. set\_quiz: that will accept a group of quiz scores in an integer array. The values will copy to the private attribute array quiz.
  3. set\_assignment: that will accept a group of assignment scores in an integer array. The values will copy to the private attribute array assignment.
  4. get\_test: will calculate and return the average of test scores in attribute array test.
  5. get\_quiz: will calculate and return the average of quiz scores in attribute array quiz.
  6. get\_assignment: will calculate and return the average of assignment scores in attribute array assignment.
  7. grading: will call the method “letter\_grading” of the super class to set the GPA score (i.e., attribute of super class).
  8. toString: will return the information in a String, which carries both the course information and the scores of tests, quizzes, and assignments. GPA information is also needed.

Evaluation:

1. The assignment is based on the materials in class. Please check the ppt file available at the class website and use the URL links to download the relevant programs/materials for study.
2. Submit all executable java files. For part 2 “program development,” you are NOT allowed to add any unnecessary attribute or method in class.
3. Do not forget to backup all your files in case your submission is lost. It will be your responsibility to resubmit the project if the instructor did not receive your submission.