**Homework 1**

For problems 1 through 4, explain why the code as shown is almost certainly not what the programmer intended, and how it should be fixed to work the way the programmer probably had in mind.

1. (10 pts) What is wrong with the following program and how should it be fixed?

1  public class MyClassA {  
2    int v = 12;  
3   
4    public MyClassA (int pV) {  
5      v = pV;  
6    }   
7   
8    public static void main (String args []) {  
9      MyClassA m = new MyClassA ();  
10   } // end main  
11 } // end class MyClassA

This is not what the programmer intended because it appears to be instantiating an object without providing an argument to set the class variable pV. To make this work properly the programmer would need to provide an argument. I would change line 9 to read (using an arbitrary argument for example sake): MyClassA m = new MyClassA (30);

Additionally, line two contains a public variable which is set by a method constructor, and therefore int v should be set to private. As a side note, variable names should be meaningful so other programmers can understand their meaning, and these variable names are not meaningful.

2. (10 pts) What is wrong with the following program and how should it be fixed?

1  public class MyClassB {  
2    int v = 12;  
3   
4    public void MyClassB (int pV) {  
5      v = pV;  
6    }   
7   
8    public static void main (String args []) {  
9      MyClassB m = new MyClassB (23);  
10   } // end main  
11 } // end class MyClassB

This is not what the programmer intended because constructors should not have return types. The compiler will misinterpret the constructor on line 4 to be a class method and not know how to instantiate the object. To fix this program, I would remove “void”, changing line 4 to read: public MyClassB (int pV) {

3. (10 pts) What is wrong with the following program and how should it be fixed?

1   public class MyClassD {  
2     public static void main (String args []) {  
3       MyClassC m = new MyClassC (23);  
4     } // end main  
5   } // end class MyClassD  
6   
7   class MyClassC {  
8     int v = 12;  
9   
10    public MyClassC (int pV) {  
11      int v = pV;  
12    }   
13   
14  } // end class MyClassC

This program has no constructor for MyClassD, which is never actually used. It also has a class MyClassC after MyClassD ends, which contains a constructor. Additionally, in the main, it is only instantiating MyClassC, which leads me to believe the MyClassD was an accident. I would change the code to the following:

1 public class MyClassC {  
2 int v = 12;  
3  
4 public MyClassC(int pV) {  
5 v = pV;  
6 }  
7  
8 public static void main(String[] args) {  
9 MyClassC m = new MyClassC (19);  
10 }//end main  
11 }//end class

4. (10 pts) What is wrong with the following program and how should it be fixed?

1   public class MyClassE {  
2     public static void main (String args []) {  
3       MyClassF m = new MyClassF (23);  
4     } // end main  
5   } // end class MyClassE  
6   
7   class MyClassF {  
8     int v = 12;  
9   
10    private MyClassF (int pV) {  
11      v = pV;  
12    }   
13   
14  } // end class MyClassF

This program is attempting to instantiate a private class within a class which does not have access because it is a different class. To fix this code, I would simply either make MyClassF public or completely do away with MyClassE and just change the entire code to MyClassF. Personally, I would change the code to read as follows:

class MyClassF {  
 int v = 12;  
  
 private MyClassF (int pV) {  
 v = pV;  
 }  
 public static void main (String args []) {  
 MyClassF m = new MyClassF (23);  
 } // end main  
} // end class MyClassF

5. (10 pts) Given all the problems identified in problems 1 through 4, explain in detail why the following code works, ie, compiles without errors or warnings.

1  public class MyClassG {  
2    public static void main (String args []) {  
3      MyClassH m = new MyClassH (23, true);  
4    } // end main  
5  } // end class MyClassG  
6   
7  class MyClassH {  
8    int v = 12;  
9   
10   public MyClassH (int x, boolean b) {  
11     this (x);  
12   }   
13   
14   private MyClassH (int pV) {  
15     v = pV;  
16   }   
17   
18 } // end class MyClassH

This compiles and runs without error because MyClassH is set to public; therefore instantiating the an object of class MyClassH within MyClassG is allowed. Both arguments (int, Boolean) are provided in the instantiating arguments. The private constructor within MyClassH is being called by this(x) which is therefore not being used by MyClassG, but by MyClassH.

6. (10 pts) Explain why the following class hierarchy is not reasonable:

* DefenseDepartment
  + General
    - Private

Private has nothing in common with general. Assuming the General class are the basic arms of the Defense Department (Army, Navy, Airforce, Marines) and private being private security firms, private does not share the same attributes as general, and therefore does not make sense for private to be a subclass of general. It makes more sense for private to be a subclass of Defense Department.

7. (10 pts) Give at least one example of a reasonable field for each of the following classes in the following class hierarchy. Be sure that the field is at the right level in the hierarchy.

* Vehicle (typeOfFuel)
  + Car (numberOfDoors)
  + Airplane (numberOfCrewRequired)
    - Passenger (numberOfPassengersCarried)
    - Fighter (numberOfMisslesCarried)
    - Bomber (numberOfBombsCarried)
  + SpaceShip (minimumEscapeVelocity)

8. (10 pts) Give at least one example of a reasonable method for each of the following classes in the following class hierarchy. Be sure that the method  is at the right level in the hierarchy. Constructors, getters and setters don't count for this problem.

* Vehicle (calculateDistanceTraveled)
  + Car (calculateTireWear)
  + Airplane (calculateMaxRange)
    - Passenger (calculateMaxWeight)
    - Fighter (calculateMaxEngagementDistance)
    - Bomber (calculateProperPayloadReleaseDistance)
  + SpaceShip (calculateAirProductionRequirements)

9. (10 pts) Are a Private and a Platoon in an encapsulation or an inheritance relationship? Explain

A private is an inheritance relationship with a platoon because a private is a member of a platoon.

10. (10 pts) Present reasonable parent and child classes for the class Tree (the biological kind). Give a short explanation for why the classes you are proposing are in good parent-child relationships.

* Tree
  + Evergreen
    - Scots Pine
    - White Fir
    - Blue Spruce
  + Deciduous
    - Maple
    - Birch
    - Ash

**Grading Rubric:**

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Meets** | **Does not meet** |
| Problem 1 | **10 points** Explains why the code as shown is almost certainly not what the programmer intended.  Explains how it should be fixed to work the way the programmer probably had in mind. | **0 points** Does not explain why the code as shown is almost certainly not what the programmer intended.  Does not explain how it should be fixed to work the way the programmer probably had in mind. |
| Problem 2 | **10 points** Explains why the code as shown is almost certainly not what the programmer intended.  Explains how it should be fixed to work the way the programmer probably had in mind. | **0 points** Does not explain why the code as shown is almost certainly not what the programmer intended.  Does not explain how it should be fixed to work the way the programmer probably had in mind. |
| Problem 3 | **10 points** Explains why the code as shown is almost certainly not what the programmer intended.  Explains how it should be fixed to work the way the programmer probably had in mind. | **0 points** Does not explain why the code as shown is almost certainly not what the programmer intended.  Does not explain how it should be fixed to work the way the programmer probably had in mind. |
| Problem 4 | **10 points** Explains why the code as shown is almost certainly not what the programmer intended.  Explains how it should be fixed to work the way the programmer probably had in mind. | **0 points** Does not explain why the code as shown is almost certainly not what the programmer intended.  Does not explain how it should be fixed to work the way the programmer probably had in mind. |
| Problem 5 | **10 points** Given all the problems identified in problems 1 through 4, explains in detail why the code works, ie, compiles without errors or warnings. | **0 points** Given all the problems identified in problems 1 through 4, does not explain in detail why the code works, ie, compiles without errors or warnings. |
| Problem 6 | **10 points** Explains why the class hierarchy is not reasonable. | **0 points** Does not explain why the class hierarchy is not reasonable. |
| Problem 7 | **10 points** Gives at least one example of a reasonable field for each of the classes.  The field is at the right level in the hierarchy. | **0 points** Does not give at least one example of a reasonable field for each of the classes.  The field is not at the right level in the hierarchy. |
| Problem 8 | **10 points** Gives at least one example of a reasonable method for each of the classes.  The method is at the right level in the hierarchy.  Does not include constructors, getters and setters. | **0 points** Does not give at least one example of a reasonable method for each of the classes.  The method is not at the right level in the hierarchy.  Includes constructors, getters and setters. |
| Problem 9 | **10 points** Explains inheritance and encapsulation correctly and in sufficient detail given the example provided. | **0 points** Does not explain inheritance and encapsulation correctly and in sufficient detail given the example provided. |
| Problem 10 | **10 points** Presents reasonable parent and child classes for the class Tree.  Gives a short explanation for why the classes you are proposing are in good parent-child relationships. | **0 points** Does not present reasonable parent and child classes for the class Tree.  Does not give a short explanation for why the classes you are proposing are in good parent-child relationships. |