Homework 1

Terrence Jackson

CMSC 335: Object-Oriented and Concurrent Programming

Professor Vergamini

Mar 19, 2024

1. (5 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
```

The constructor for MyClassA is implemented in lines 4-6. In this implementation, the constructor takes an integer parameter. In line 9, the main function creates an instance object of MyClassA. However, the programmer did not include the integer parameter when they instantiated this new object. There is no default constructor that takes no arguments. I believe the intention was for line 9 to be written as follows:

```
MyClassA m = new MyClassA(12);
```

2. (5 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
```

The programmer has attempted to define a constructor for MyClassA in lines 4-6. However, in this program the constructor is declared as a regular method due to the inclusion of the return type “void”. To be a proper constructor, there should be no return type. I believe the programmer’s intention was to write line 4 as follows:

```
public MyClassB(int pV) {
```

3. (5 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
```

In this program, the problem lies in line 11. Inside the constructor for MyClassC, the integer variable v is declared and initialized. This makes it a local variable to the constructor. I believe this is a mistaken attempt to use the variable of the same name that was declared and initialized on line 8. In order to fix this mistake, line 11 should remove the data type and be written as follows:

```
    v = pV;
```

4. (5 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
```

The constructor for MyClassF is implemented in lines 10-12. MyClassE attempts to use this constructor in its main function in line 3. However, the constructor was defined as “private”, so it is not accessible outside of MyClassF. I believe the programmer meant for this constructor to be accessible by MyClassE:

```
public MyClassF (int pV) {
```

5. (5 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
```

In problem 1, the error came from arguments not being passed to a constructor that required them. In this program, each constructor is called with the appropriate number of arguments. In line 3, the public constructor for MyClassH is called. This constructor is defined in lines 10-12 and takes two arguments. In line 11, the private constructor is called. This constructor is defined in lines 14-16 and takes one argument. In both cases, the calls to the constructor pass in the appropriate number of arguments.

In problem 2, the error came from incorrectly defining the constructor. In this program, both the public and private constructors are correctly written without a return type.

In problem 3, the problem came from redeclaring a variable. In this program, the class variable v is declared and initialized in line 8. That same class variable v is then used in line 15. The variable is not redeclared in line 15 as a local variable.

In problem 4, a private constructor was declared, but the programmer needed to use it outside of the class. In this program, MyClassH has both a private and a public constructor. The public one is used by MyClassG. The private one is only called within MyClassH.

6. (5 pts) Explain why the following class hierarchy is not reasonable:

- `DefenseDepartment`
    - `General`
        - `Private`

While it is an accurate hierarchy, it is not a class hierarchy. A class hierarchy consists of "is-a" relationships. A General is not a kind of Defense Department, nor is a Private a kind of General.

7. (5 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`
    - `Car`
    - `Airplane`
        - `Passenger`
        - `Fighter`
        - `Bomber`
    - `SpaceShip`

Vehicle	String engineType
Car	int numberOfDoors
Airplane	int numberOfWindows
Passenger	int numberOfSeats
Fighter	float percentDamaged
Bomber	int numberOfBombs
SpaceShip	Boolean artificialGravityOn

8. (5 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
  - Car
  - Airplane
    - Passenger
    - Fighter
    - Bomber
  - SpaceShip

Vehicle	public double calculateSpeed(double distance, double time)
Car	public double calculateFuelConsumption(double distance)
Airplane	public Boolean takeOff(double runwayLength) // return success/failure
Passenger	public int boardPassenger(String passengerName) // return seat number
Fighter	public Boolean engageTarget(String targetName) // return success/failure
Bomber	public Boolean releaseBomb(String targetLocation) // return success/failure
SpaceShip	public Boolean fuelEngines(double amount) // return T/F fuel tank full

9. (5 pts) Please provide an example of an encapsulation and an inheritance relationship? Explain

Encapsulation is about bundling together fields and methods into a class and abstracting away direct access to the fields of the class. For example, the Airplane class might have fields such as speed, fuelLevel, and engineStatus. However, the class should have getter and setter methods for each of these fields. Instead of directly accessing fuelLevel and being able to change it, a programmer will have to use the getFuelLevel() and setFuelLevel() methods to interact with the field.

An inheritance relationship is like the problem above, where Passenger inherits from Airplane. Passenger is a type of Airplane, and therefore it will inherit the fields and methods such as numberOfWindows and takeoff(). However, Passenger will also have its own unique fields and methods such as numberOfSeats and listPassengers().

10. (5 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.

- `Plant`
    - `Bush`
    - `Tree`
        - `Birch`
        - `Spruce`
        - `Apple`
    - `Flower`

Plant is the parent class of Tree because a tree is a kind of plant. A tree is a specific kind of plant that has its own unique attributes and behavior. The subclasses of Tree are all unique kinds of trees: birch trees, spruce trees, and apple trees. All these subclasses will have things in common such as a field for leaf color or a method for photosynthesis. Additionally, the subclasses will have their own unique fields and methods. For example, apple trees will growFruit() and have an appleColor field, whereas birch and spruce do not.