Homework 2

Terrence Jackson

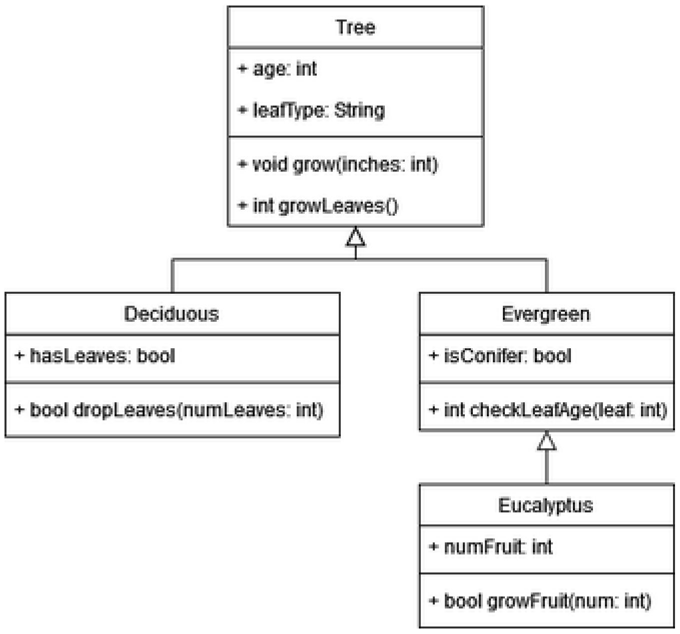
CMSC 335: Object-Oriented and Concurrent Programming

Professor Vergamini

Apr 2, 2024

1. (5 pts) What are the diagrams defined in the UML Standard. Give a one or two sentence description of each one. Please provide a small example of a UML diagram.

There are two main kinds of UML diagrams: structure diagrams and behavior diagrams (Fakhroutdinov, 2013). There are several kinds of behavior diagrams, including use case, activity, state machine, interaction, sequence, communication, timing, and interaction overview diagrams (Fakhroutdinov, 2013). In this class, we are mainly focusing on class diagrams, but there are several other kinds of structure diagrams, including object, package, composite structure, component, deployment, and profile diagrams (Fakhroutdinov, 2013). A class diagram displays the arrangement of a system or component (Fakhroutdinov, 2013). It shows off the connections between various classes and interfaces, including their attributes and methods (Fakhroutdinov, 2013).



2. (5 pts) Given the following code, how should the toString methods in the classes H2ClassA and H2ClassB be written to give the indicated output and take advantage of the natural toString method in H2ClassB?

```
import java.util.ArrayList;

public class H2ClassA {
    ArrayList <H2ClassB> list = new ArrayList <H2ClassB> ();

    public static void main (String args []) {
        H2ClassA y = new H2ClassA ();
        int [] v = {4, 3, 7, 5, 99, 3};
        for (int m: v)
            y.list.add (new H2ClassB (m));
        System.out.println (y);
    } // end main

} // end class H2ClassA

class H2ClassB {
    int x;
    H2ClassB (int a) { x = a;}
} // end H2ClassB
```

```
OUTPUT:
4 3 7 5 99 3
```

In class H2ClassB, the toString() method should return a string representation of the integer x variable. This uses the natural toString() method from the Integer class:

```
@Override
public String toString() {
    return Integer.toString(x); // natural toString()
}
```

In class H2ClassA, the toString() method should iterate over the items in the ArrayList of H2ClassB items. I chose to use a StringBuilder object to combine each list element together into one string (*The stringbuilder class*). StringBuilder objects are modifiable and are more efficient than concatenating String objects in a loop (*The stringbuilder class*). Once all the list elements have been added to the StringBuilder, I used it's natural toString() method to return the full String (*The stringbuilder class*).

```
@Override
public String toString() {
    StringBuilder listString = new StringBuilder();
    for (H2ClassB item : this.list) { // Iterate over items in ArrayList
        listString.append(item); // Append each item to the StringBuilder
        listString.append(" "); // Add a space between items
    }
    return listString.toString(); // Convert StringBuilder to String
}
```

3. (5 pts) How can the following code be corrected? Give at least two good answers.

```
public class H2ClassC {
    H2ClassC (int a) {}
} // end class H2ClassC

class H2ClassD extends H2ClassC{
} // end class H2ClassD
```

H2ClassD does not explicitly call any constructor, so it automatically calls the default constructor of its' parent class. H2ClassC is the parent of H2ClassD, but H2ClassC does not have a default constructor. Either H2ClassC should implement a default constructor, or H2ClassD should implement a constructor that utilizes H2ClassC's parameterized constructor.

Solution 1: Implement a default constructor for H2ClassC

```
H2ClassC() { int a = 5; }
```

Solution 2: Implement a constructor for H2ClassD

```
class H2ClassD extends H2ClassC {
    H2ClassD(int b) {
        super(b);
    }
} // end class H2ClassD
```

Class H2ClassC's parameterized constructor takes one integer argument. However, the constructor does not do anything with the argument passed into it. Regardless of what value is passed in, the argument does not affect the behavior of the function. Either the constructor should not require this argument, or it should initialize an instance variable using the argument.

Solution 1: Implement a constructor for H2ClassC that doesn't require the argument

```
H2ClassC() { int a = 10; }
```

Solution 2: Use the argument

```
public class H2ClassC {
    int a;

    H2ClassC(int a) {
        this.a = a;
    }
} // end class H2ClassC
```

4. (5 pts) Why does the following code give a compiler error? How should it be fixed?

```
1  public class H2ClassE {
2      int x, y, z;
3
4      H2ClassE (int a) {
5          x = a;
6          this (5, 12);
7      }
8
9      H2ClassE (int b, int c) {
10         y = b;
11         z = c;
12     }
13 } // end class H2ClassE
```

H2ClassE has two constructors. The first constructor takes one integer argument. It sets the class attribute x with the argument a, then calls H2ClassE's second constructor. The second constructor takes two integer arguments, then sets y and z. The error is caused by line 6, because a call to a different constructor needs to be the first statement in a constructor. The solution is to switch lines 5 and 6, which ensures the call to the second constructor happens before any other statements:

```
H2ClassE (int a) {
    this (5, 12);
    x = a;
}
```

5. (5 pts) What is wrong with the following declaration? How should it be fixed?

```
public static final int myNumber = 17.36;
```

The variable myNumber is declared as an integer, but is being initialized with a floating point value. This could be solved by declaring the variable as a double or by casting the floating point value to an integer value. While casting to an integer does remove the error, it truncates the decimal part of the value, effectively rounding it down to 17. This solution is only acceptable if the fractional part of the original value was not significant.

```
public static final double myNumber = 17.36; // as a double
public static final int myNumber = (int) 17.36; // cast to int
```

6. (5 pts) What is wrong with the following code? How should it be fixed?

```
1 public class H2ClassG {
2     final int x;
3
4     H2ClassG () {}
5     H2ClassG (int a) {x = a;}
6 } // end class H2ClassG
```

The class H2ClassG has a final integer attribute, x. The final keyword is used to declare variables that cannot be reassigned after being initialized. This means that x must be initialized exactly once, either when it is declared or in a constructor. In the parameterized constructor on line 5, the value of x is initialized. However, in the constructor on line 4, the value of x is not initialized. This could lead to unexpected behavior if the default constructor is used to create an object and then the x attribute of the object is accessed. This can be solved by initializing x in the default constructor:

```
    H2ClassG () { x = 5; }
```

7. (5 pts) What is wrong with the following code? How should it be fixed?

```
1 public class H2ClassH {
2     final int x;
3
4     int H2ClassH () {
5         if (x == 7) return 1;
6         return 2;
7     } // end
8 } // end class H2ClassH
```

In line 4, there is a method that is named following the convention for a constructor, but it returns an integer value. A proper constructor would not return any value, so I believe this method should be renamed. Renaming it will make it clear that it is not meant to be a constructor.

In line 2, a final integer variable is declared but not initialized. In line 5, the variable x is accessed. Even if x were initialized elsewhere in the code, it is a class member variable and won't be automatically initialized to a default value. So, it needs to be initialized explicitly, either during declaration or in a constructor. Without initializing this variable, it is not possible to accurately compare its equality to 7. One solution is to add a constructor that initializes x.

```
public class H2ClassH {
    final int x;

    H2ClassH () { x = 7; } // constructor to init x

    int someMethod() {
        if (x == 7) return 1;
        return 2;
    } // end someMethod
} // end class H2ClassH
```

8. (5 pts) What is wrong with the following code? x should be given a value of 24. What are two ways this can be legally accomplished?

```
1 public class H2ClassI {
2     final int x;
3
4     public static void main (String args []) {
5         H2ClassI h = new H2ClassI ();
6         h.x = 24;
7     } // end main
8 } // end class H2ClassI
```

In line 2, the final integer field x is declared. Because it is a final variable, it needs to be initialized either in the declaration or in the constructor of H2ClassI. In line 5, an object of type H2ClassI is created. Then on line 6, there is an attempt to assign a value to to the object's x field. It is not possible to assign a value to a final field in this way. It would be better to initialize the value in the declaration, or create a constructor to initialize it.

Solution 1: Initialize x in declaration

```
public class H2ClassI {
    final int x = 24; // init in dec

    public static void main(String args[]) {
    H2ClassI h = new H2ClassI();
    } // end main
} // end class H2ClassI
```

Solution 2:Initialize x in constructor

```
public class H2ClassI {
    final int x;

    H2ClassI() { x = 24; } // init in constructor

    public static void main(String args[]) {
        H2ClassI h = new H2ClassI();
    } // end main
} // end class H2ClassI
```

9. (5 pts) What is wrong with the following Swing code? Give two effective ways to fix it.

```
1  import javax.swing.*;
2  import java.awt.event.*;
3
4  public class H2ClassJ extends JFrame {
5     public static final long serialVersionUID = 22;
6
7     public H2ClassJ () {
8           addMouseListener (new MouseListener () {
9           public void mouseClicked (MouseEvent e) {}
10          });
11 } // end constructor
12
13 } // end class H2ClassJ
```

In line 4, H2ClassJ is declared as a child class because it extends the parent class JFrame. In line 7, a constructor for the class is declared. Inside the constructor, in line 8, the method addMouseListener is called. The argument that is passed into addMouseListener is a new MouseListener. However, MouseListener is an interface, which means that it must be implemented before it can be instantiated. In order to implement MouseListener, the method mouseClicked is implemented in line 9. The problem is that MouseListener requires all of its methods be implemented. In addition to mouseClicked, the methods mousePressed, mouseReleased, mouseEntered, and mouseExited need to be implemented. One solution is to implement all these methods just as mouseClicked was on line 9:

```
8  addMouseListener(new MouseListener() {
9     public void mouseClicked(MouseEvent e) {}
10    public void mousePressed(MouseEvent e) {}
11    public void mouseReleased(MouseEvent e) {}
12    public void mouseEntered(MouseEvent e) {}
13    public void mouseExited(MouseEvent e) {}
14 });
```

Another solution is to use an abstract class instead of an interface. In the Java documentation, it states that MouseAdapter is "An abstract adapter class for receiving mouse events. The methods in this class are empty. This class exists as convenience for creating listener objects." (*MouseAdapter*, 2024). The goal of line 8 seems to be to create a listener object, so this abstract class would serve the purpose much better. This way, the programmer can choose which methods to implement:

```
8  addMouseListener(new MouseAdapter() {
9     public void mouseClicked(MouseEvent e) {}
10 });
```

One more thing to note is that line 9 implements the mouseClicked method, but does not add any functionality to the method. It would be better to implement any MouseAdapter methods in a way that would add to the functionality of the program, otherwise it may be better to leave the method implementation out:

```
8  addMouseListener(new MouseAdapter() {});
```

10. (5 pts) What is incorrect in the following FX GUI code?

```
1 import javax.javafx.*;
2
3 public class H2ClassK {
4     submit.setOnAction((ActionEvent e) → {
5           label.setText("A comment");
6     });
7 } // end class H2ClassK
```

There are several things missing from this code. Following the example in the Getting Started with JavaFX guide, there should be more explicit imports directly from javafx, for example: "`import javafx.event.ActionEvent;`" rather than importing from `javax.javafx` (*Release: Javafx 2.2.40*, 2013). The next thing missing is the variable declarations for `submit` and `label`. In line 4, `submit` is used to call `setOnAction`, however, this is the first time the variable appears in the code. Similarly, `label` is used in line 5 without having been declared beforehand. Based on the context given, these should be javafx.scene.control.Button and javafx.scene.control.Label objects, respectively. There is also an error in line 5, where the quotation marks are not matched. It should be `"A comment"` with matching quotation marks. Without more context, it is difficult to assume what the programmer intended. Following the example in the Getting Started with JavaFX guide, the imports have been adjusted, variables have been declared and initialized, and the quotation marks have been fixed. What follows is a potential solution to implement behavior closer to what I believe the code was attempting to create:

```java
import javafx.application.Application;
import javafx.event.ActionEvent;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.layout.VBox;
import javafx.stage.Stage;

public class H2ClassK extends Application {
   public static void main(String[] args) {
         launch(args);
   }

   @Override
   public void start(Stage primaryStage) {
         Button submit = new Button();
         submit.setText("Submit");
         Label label = new Label();

         submit.setOnAction((ActionEvent e) -> {
               label.setText("A comment");
         });

         VBox root = new VBox();
         root.getChildren().add(submit);
         root.getChildren().add(label);
         primaryStage.setScene(new Scene(root, 300, 250));
         primaryStage.show();
   }
} // end class H2ClassK
```

References

Fakhroutdinov, K. (2013, November 25). *UML 2.5 diagrams overview*. UML Diagrams - overview, reference, and examples. https://www.uml-diagrams.org/uml-25-diagrams.html

MouseAdapter (Java Platform SE 8). (2024, January 8). https://docs.oracle.com/javase/8/docs/api/java/awt/event/MouseAdapter.html

Release: Javafx 2.2.40. Getting Started with JavaFX: Hello World, JavaFX Style | JavaFX 2 Tutorials and Documentation. (2013, August 30). https://docs.oracle.com/javafx/2/get_started/hello_world.htm

The stringbuilder class. The StringBuilder Class (The JavaTM Tutorials > Learning the Java Language > Numbers and Strings). (n.d.). https://docs.oracle.com/javase/tutorial/java/data/buffers.html