Homework 2

1. (10 pts) What are the diagrams defined in the UML Standard. Give a one or two sentence description of each one.

In the UML Standard, “there are two major kinds of diagram types: structure diagrams and behavior diagrams” (Object Management Group, 2017, P. 727, Para. 1). Structure diagrams are for static elements. These are very similar to blueprints for a building: they may show all the rooms, the objects inside the room, and possibly the elements and “functions” of the objects and rooms. These diagrams often show relationships, as well. Behavior diagrams, on the other hand, are for the dynamic elements. These diagrams show how the related objects perform or interact. To stay with the blueprint example, a behavior diagram would illustrate why and/or how the shower gets hot/cold when someone flushes a toilet. I’ve seen some online sources giving a third diagram type, interaction diagram, but according to the standard, this is a subcategory of behavior diagrams (Object Management Group, 2017, pg. 727, Figure A.5).

2. (10 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?

```java
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
```

To produce the same output, H2ClassB’s toString method should be written as “`public String toString() { return String.valueOf(x); }`.” This takes advantage of the “natural” toString method of x. The toString method of H2ClassA requires a bit more work to match the desired output and should be written as :

```
     public String toString() {
       StringBuffer output = new StringBuffer();
       for (H2ClassB b: list) {
         output.append(b.toString());
         if (list.indexOf(b) < list.size()) {
           output.append(“ “);
         }
       }
       return output.toString();
     }
```

While it is possible to simplify this to “`public String toString() { return list.toString(); }`,” doing so surrounds the output in brackets and places commas between each element. In other words, it would output “`[4, 3, 7, 55, 99, 3]`.”

3. (10 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
```

One potential way to correct the code would be to add a constructor with a single int parameter to the H2ClassD class. For example, “H2ClassD (int a) { super(a); }.” Another potential way to correct the code would be to add an argument-less constructor to the H2ClassC class, like “H2ClassC () {}.” This type of constructor is implicitly defined if and only if there are no explicitly defined constructors, like H2ClassD. Since H2ClassC has a constructor explicitly defined, this “empty” constructor would need to be present to take advantage of H2ClassD’s implicit constructor. Also, if H2ClassC and H2ClassD are in different packages, H2ClassC’s constructors would also need to be prepended with “public.”

4. (10 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
```

The code gives a compiler error because the statements in the H2ClassE(int) constructor are not in the proper order. Constructor statements must follow a predefined order: calls to super, calls to other constructors in class, and, finally, all other statements. That said, the code can be fixed by switching lines 5 and 6.
The fixed constructor:

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

5. (10 pts) What is wrong with the following declaration? How should it be fixed?

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

The declaration contains a semantic error and a conventional error. The semantic error is a type mismatch error which can be resolved by either changing the value to an integer or by changing the type to float or double. The conventional error can be ignored as it has no effect on function, but should be resolved to adhere to industry standards. This error can be resolved by capitalizing all letters in the name and separating words with underscores, like "MY_NUMBER" (see http://java.sun.com/docs/codeconv/html/CodeConvention.doc8.html or https://google.github.io/styleguide/javaguide.html#S5.2.4-constant-names).

6. (10 pts) What is wrong with the following code? How should it be fixed?

```
public class H2ClassG {
  final int x;

  H2ClassG () {}
  H2ClassG (int a) {x = a;}
} // end class H2ClassG
```

Since the instance variable “x” is declared final, it must be assigned a value. While this is done in the H2ClassG (int) constructor, it was not done in the H2ClassG () constructor. This could be fixed by performing any one of three things: (1) adding and assignment statement to the argument-less constructor, like “x = 10;”, (2) removing the “final” keyword from the variable declaration, or (3) removing the assignment statement from the H2ClassG(int) constructor and appending the variable declaration with an assignment to read, as an example, “final int x = 10.”

7. (10 pts) What is wrong with the following code? How should it be fixed?

```
public class H2ClassH {
  final int x;

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

The code has a syntax error and a conventional error. The syntax error is an uninitiated variable error which can be resolved by assigning a value to “x” at its declaration, similar to resolution 3 of problem 6. Another option is to remove the “final” keyword, add an int parameter to the “int H2ClassH ()” method, and assign the value of the parameter to the variable “x”. The conventional error can be resolved by simply making the first character in the H2ClassH() method’s name lowercase, like “int h2ClassH ().”

8. (10 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
```

The code will fail to compile because the instance variable “x” of H2ClassI is labeled “final” and is not assigned a variable at declaration nor in a constructor. One way would be to define the variable at declaration, like “final int x = 24;”. Another way would be to define the argumentless constructor and define “x” within it, like:

```
        public H2ClassI () {
          x = 24;
        }
```

With either of these methods, line 6, “h.x = 24;”, must be removed. This is because the variable cannot be changed after it has been assigned a value due to the “final” label.

9. (10 pts) What is wrong with the following code? Give two effective ways to fix it.

```
import javax.swing.*;
import java.awt.event.*;

public class H2ClassJ extends JFrame {
  public static final long serialVersionUID = 22;

  public H2ClassJ () {
    addMouseListener (new MouseListener () {
      public void mouseClicked (MouseEvent e) {}
    });
  } // end constructor

} // end class H2ClassJ
```

The code will fail to compile because the anonymous object created by “new MouseListener()” does not implement all of the required methods. One way to resolve this would be to define the missing methods, namely mouseEntered(MouseEvent e), mouseExited(MouseEvent e), mousePressed(MouseEvent e), mouseReleased(MouseEvent e). This is obviously a lot more code than the programmer was wishing to use. Luckily, Java has included another resolution: the MouseAdapter class. The MouseAdapter class defines empty methods for every abstract method in the MouseListener interface (Oracle, 2020). This is extremely useful with anonymous classes in which only one method is needed.

10. (10 pts) Why does the following code give a compiler warning? (Use javac -Xlint) How should it be fixed?

```
1 import javax.swing.*;
2
3 public class H2ClassK {
4   String [] sa = {"a", "b", "c"};
5   JComboBox jcbA = new JComboBox (sa);
6 } // end class H2ClassK
```

The code gives a compile warning because the element type is not defined for the JComboBox element. JComboBox is a generic class, meaning it can be initialized using any type of element. The elements' type must be defined within angle brackets, or "<>". Since the elements of the array "sa" are String Objects, line 5 should be changed to read "JComboBox<String> jcbA = new JComboBox<String> (sa);" or "JComboBox<String> jcbA = new JComboBox<> (sa);". The second angle brackets can be left empty, because the type is defined with the object type. It is helpful to include the type within the initialization when it is separate from the declaration. For example:

```
        JComboBox<String> jcbA;
        # some other class variables

        # argument-less constructor
        public H2ClassK () {
                jcbA = new JComboBox<String> (
                        new String{}{"a", "b", "c"});
        }

        public H2ClassK (String [] input) {
                jcbA = new JComboBox<String> (input);
        }
```

Grading Rubric:

Attribute	Meets	Does not meet
Problem 1	**10 points** Gives a one or two sentence description of each standard UML diagram.	**0 points** Does not give a one or two sentence description of each standard UML diagram.
Problem 2	**10 points** Explains how 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.	**0 points** Does not explains how 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.
Problem 3	**10 points** Provides at least two good answers explaining how the code can be corrected.	**0 points** Does not provide at least two good answers explaining how the code can be corrected.
Problem 4	**10 points** Explains why the code gives a compiler error. Explains how the code should be fixed.	**0 points** Does not explain why the code gives a compiler error. Does not explain how the code should be fixed.
Problem 5	**10 points** Explains what is wrong with the declaration. Explains how the code should be fixed.	**0 points** Does not explain what is wrong with the declaration. Does not explain how the code should be fixed.
Problem 6	**10 points** Explains what is wrong with the code. Explains how the code should be fixed.	**0 points** Does not explain what is wrong with the code. Does not explain how the code should be fixed.
Problem 7	**10 points** Explains what is wrong with the code. Explains how the code should be fixed.	**0 points** Does not explain what is wrong with the code. Does not explain how the code should be fixed.
Problem 8	**10 points** Explains what is wrong with the code. Explains two ways x could be given a values of 24 legally.	**0 points** Does not explain what is wrong with the code. Does not explain two ways x could be given a values of 24 legally.
Problem 9	**10 points** Explains what is wrong with the code. Explains 2 effective ways the code could be fixed.	**0 points** Does not explain what is wrong with the code. Does not explain 2 effective ways the code could be fixed.
Problem 10	**10 points** Explains why the code gives a compiler warning. Explains how it should be fixed.	**0 points** Does not explain why the code gives a compiler warning. Does not explain how it should be fixed.

References

Object Management Group. (2017). *Unified Modeling Language* (version 2.51). Retrieved from https://www.omg.org/spec/UML/2.5.1/PDF.

Oracle. (2020, July 09). *MouseAdapter (Java Platform SE 8)*. Java™ Platform, Standard Edition 8 API Specification. https://docs.oracle.com/javase/8/docs/api/java/awt/event/MouseAdapter.html