YORK UNIVERSITY
AUGUST 2022 FINAL EXAM

EECS 3311
SOFTWARE DESIGN
Duration - 3 hours
Aids: None

Good Luck!

Question 1. [18 MARKS]

The code below is an attempt to use the **iterator** design pattern to implement a `Digits` class that allows us to iterate through the digits of a number (from the lowest, or rightmost digit, to the highest, or leftmost digit). For example, the following for-loop

```
for (int d : new Digits(2023)) {
    System.out.println(d);
}
```

would ideally produce the following output:

```
Digits of 2023
3
2
0
2
```

Below is the attempted implementation of the `Digits` and `DigitsIterator` classes.

```
import java.util.Iterator;
class DigitsIterator implements Iterator<Integer> {

    private int value;
    public DigitsIterator(int value) {
        this.value = value;
    }
    public boolean hasNext() {
        return value >= 10;
    }
    public Integer next() {
        return this.value % 10;
    }
}

class Digits implements Iterable<Integer> {
    private int value;
    public Digits(int value) {
        System.out.println("Digits of " + value);
        this.value = value;
    }
    public Iterator<Integer> iterator() {
        return new DigitsIterator(this.value);
    }
}
```

However, as you may have noticed, the above code is not completely correct. Your job in this question is to fix it. Answer the following questions:

Part (a) [4 MARKS]

With the above buggy code, what is the output of the following for-loop?

```
for (int d : new Digits(2023)) {
    System.out.println(d);
}
```

Part (b) [4 MARKS]

With the above buggy code, what is the output of the following for-loop?

```
for (int d : new Digits(9)) {
    System.out.println(d);
}
```

Part (c) [10 MARKS]

Fix the above code by rewriting the incorrectly implemented methods in the space provided below. You should keep the amount of change as minimal as possible.

Question 2. [15 MARKS]

In each of the scenarios given below, name the design pattern that would be the most appropriate to implement.

(a) You are designing an application with a Graphical User Interface (GUI). You want to implement a menu structure, where each menu element is either (1) an action that the user can perform, or (2) a submenu containing other menu elements.

(b) Your application needs a logger class to maintain a history of some debugging information (e.g., memory usage). Your logger class will be used by many other objects within your application, but you want to ensure that they are all sharing **one** instance of your logger class. In other words, you want to avoid the situation where multiple logger objects are (accidentally or deliberately) instantiated.

(c) You an are designing a fast food ordering system that receives orders from customers through a smartphone app. Your ordering system will queue up orders as they are received, and should make it easy to perform various generic operations on the orders (e.g., repeat an order, cancel an order).

(d) You are working on a backup utility that supports several algorithms for compressing the user's backup files. You want to make it easy to extend your utility to support new compression algorithms; i.e., adding a new compression algorithm should be as simple as defining a new class, with minimal or no changes to the existing code.

(e) You are designing a web application that supports several layouts (e.g., smartphone, tablet, and desktop layouts). You want to define a class so that when the user clicks on a link to view a page, the web application will invoke a method in your class to generate a template page based on the user's current layout. Your web application can then insert the content into the template page and send it to the user's device.

Question 3. [16 MARKS]

Recall the code below from our lecture; it implements the bubblesort sorting method for an array of integers of length n. For each of the lines that indicate the number of marks, please do complete the required part of the JML contract.

```
public class BubbleSort {

    // PRE: arr is not null
    // POST: arr is sorted in non-decreasing order

    public static void sort(int [] arr) {
        // The variable n declared here may be used in the JML contracts
        // required by this exercise. Please do not alter this declaration.
        //@ final ghost int n = arr.length;

        // The first part of the outer loop invariant, specifies the bounds for i
        // (2 marks) Please write below the first part of the outer loop invariant
        //
        //
        // The second part of the outer loop invariant specifies that elements
        // up to i are sorted
        // (2 marks) Please write below the second part of the outer loop invariant
        //
        //
        // The following statement declares a formula that decreases for each iteration
        // of the outer loop. It is used by the prover to show the termination.
        // (2 marks) Please write below the required formula using
        // the JML "decreasing" specification.
        //
        //
        for (int i = 0; i < arr.length; i++) {

            // The first part of the inner loop invariant, specifies the bounds for j.
            // (2 marks) Please write below the first part of the inner loop invariant
            //
            //
            // The second part of the inner loop invariant specifies that
            // j-th element is always the largest
            // (2 marks) Please write below the second part of the inner loop invariant
            //
            //
            // (code continues on the next page)
            // The third part of the inner loop invariant specifies that
            // elements up-to i remain sorted
            // (2 marks) Please write below the third part of the inner loop invariant
            //
            //
            // Please do not change the following annotation.
            //@ decreasing j;
            for (int j = arr.length-1; j > i; j--) {
                if (arr[j-1] < arr[j]) {
                    int tmp = arr[j];
                    arr[j] = arr[j-1];
                    arr[j-1] = tmp;
                }
            }
        }
    }
}
```

In the space below, please write the JML specifications for the precondition and postcondition of the `sort` method.

(2 marks) PRECONDITION

(2 marks) POSTCONDITION

Note: Some useful JML keywords might be: ensures, requires, forall, old.

Question 4. [10 MARKS]

We need to test the GET method of /api/v1/elapsed endpoint which takes two time parameters t1, t2 in 12-hour format and returns elapsed time in minutes starting from t1 and ending at t2. Some examples of values passed in would be {t1: ”5:53 AM”, t2: ”8:54 AM”} (the elapsed time would be 181 minutes). Here is the the code:

```
int calculate_elapsed(String t1, String t2) {
    // B\In the code below, the method hours computes hours in a time string
    // For example, hours("5:53 AM") returns the integer 5.
    // Likewise, minutes method returns minutes: minutes("5:53 AM") returns 53
    // Finally, ampm returns a string, either "AM" or "PM": ampm("5:51 AM") returns "AM"
    // You may assume all three methods have been implemented and tested.
    int h1 = hours(t1);
    int m1 = minutes(t1);
    String ap1 = ampm(t1);
    int h2 = hours(t2);
    int m2 = minutes(t2);
    String ap2 = ampm(t2);


    if (h1 == 12)
        h1 = 0;
    if (h2 == 12)
        h2 = 0;
    if (ap1.equals("PM"))
        h1 = h1 + 12;
    if (ap2.equals("PM"))
        h2 = h2 + 12;
    if (m2 < m1) {
        m2 = m2 + 1;
        h2 = h2 - 1;
    }
    if (h2 < h1)
        h2 = h2 + 24;
    elapsed = m2 - m1 + 60 * (h2 - h1);
    return elapsed;
}
```

Here are the test cases we have so far:
{t1: ”12:00 PM”, t2: ”12:40 PM”}
{t1: ”9.57 PM”, t2: ”11:40 PM”}
{t1: ”5:00 PM”, t2: ”4:00 AM”}

If we want to guarantee every branch coverage, for the scenario when elapsed time is less than 24 hours, from the following list, which test case do we need we add? Circle only one choice. Make sure to fill in the appropriate entry in the associated bubblesheet.

A. {t1: ”12:10 PM”, t2: ”12:23 PM”}

B. {t1: ”8:01 AM”, t2: ”12:40 PM”}

C. {t1: “7:22 PM”, t2: “11:34 PM”}

D. {t1: ”5:05 PM”, t2: ”4:33 AM”}

Question 5. [10 MARKS]

Draw a class diagram representing the relationship between parents and children, considered as persons. Take into account that a person can have both a parent and a child. Annotate associations with roles and multiplicities.

Consider the following outline of code:

```
interface Blah { ... }
class B implements Blah { ... }
abstract class Clem { ... }
class C extends Clem { ... }
class Dreft { ... }
class D extends Dreft { ... }
```

Please answer the questions 6 to 11, which are based on the given code above.

Question 6. [4 MARKS]

Consider the code line `Blah x = new B();`. From possible answers below, select the correct answer. Fill the corresponding item in the bubblesheet.

A. The give code line compiles.

B. The given code line does not compile.

Question 7. [4 MARKS]

Consider the code line `B y = new Blah();`. From possible answers below, select the correct answer. Fill the corresponding item in the bubblesheet.

A. The give code line compiles.

B. The given code line does not compile.

Question 8. [4 MARKS]

Consider the code line `z = (B) new Blah();`. From possible answers below, select the correct answer. Fill the corresponding item in the bubblesheet.

A. The give code line compiles.

B. The given code line does not compile.

Question 9. [4 MARKS]

Must `B` implement all the unimplemented methods of `Blah`? From possible answers below, select the correct answer. Fill the corresponding item in the bubblesheet.

A. Yes.

B. No. If it doesn't, we must not make an instance of B.

C. No. If it doesn't, we can make an instance of B, but we must not call the unimplemented methods.

D. `Blah` cannot have any unimplemented methods.

Question 10. [5 MARKS]

Can `Blah` implement any methods according to our course conventions? From possible answers below, select the correct answer. Fill the corresponding item in the bubblesheet.

A. It may declare static methods but not instance methods.

B. It may declare instance methods but not static methods.

C. It may declare both static methods and instance methods.

D. No. It cannot declare static methods or instance methods.

Question 11. [5 MARKS]

Can `Clem` declare any variables? From possible answers below, select the correct answer. Fill the corresponding item in the bubblesheet.

A. It may declare static variables but not instance variables.

B. It may declare instance variables but not static variables.

C. It may declare both static variables and instance variables.

D. No. It cannot declare static variables or instance variables.

Question 12. [5 MARKS]

Cosider the code

```
public class MyProgram {
   public static void throwit() {
      throw new Exception();
   }
   public static void main(String args[]) {
      try {
            System.out.println("One");
            throwit();
            System.out.println("Two");
      } finally {
            System.out.println("Three");
      }
   }
}
```

From the following, select the correct order in which messages are printed. Please complete the corresponding bubblesheet item.

A. One, Two, Three, Exception ...

B. Exception ..., One, Two

C. One, Three, Exception ...

D. One, Two, Exception ...

[Use the space below for rough work. This page will ***not*** *be marked, unless you clearly indicate the part of your work that you want us to mark.]*

[Use the space below for rough work. This page will ***not*** *be marked, unless you clearly indicate the part of your work that you want us to mark.]*