Preview Test: CSSE2002 / CSSE7023 Semester One Deferred Exam 2020

Test Information

Description	CSSE2002 Programming in the Large / CSSE7023 Advanced Software Engineering Exam Duration: 120 minutes + 30 minutes additional time Materials permitted include one physical sheet of A4 paper (double-sided) of typed or hand-written notes, four sheets of blank A4 paper for rough work and planning, writing implements (pen, pencil, eraser and/or ruler), and a water bottle.
Instructions	
Timed Test	This test has a time limit of 2 hours and 30 minutes.This test will save and be submitted automatically when the time expires. Warnings appear when **half the time**, **5 minutes**, **1 minute**, and **30 seconds** remain. *[The timer does not appear when previewing this test]*
Multiple Attempts	Not allowed. This Test can only be taken once.
Force Completion	This Test can be saved and resumed at any point until time has expired. The timer will continue to run if you leave the test.
	Your answers are saved automatically.

QUESTION 1

0 points Save Answer

Please use this space to specify any assumptions you have made in completing the exam. Clearly indicate to which questions those assumptions relate. You may also include queries you may have made with respect to a particular question, should you have been able to "raise your hand" in an examination room.

QUESTION 2

6 points Save Answer

What would be the output when the `main()` method is executed?

```java
import java.util.List;
import java.util.ArrayList;

public class Q2 {
    public static void main(String[] args) {
        List<Integer> threes = new ArrayList<>();
        List<Integer> twos = new ArrayList<>();
        int sum = 0;
        boolean test = true;
        for (int i = 0; i < 20; i++) {
            if (i % 3 == 0) {
                threes.add(i);
            } else if (i % 2 == 0) {
                twos.add(i);
            } else {
                if (test) {
                    sum += i;
                }
                test = !test;
            }
        }

        System.out.println("threes: " +  threes);
        System.out.println("twos: " +  twos);
        System.out.println("sum: " + sum);
    }
}
```

```
threes: [0, 3, 6, 9, 12, 15, 18]
twos: [2, 4, 8, 10, 14, 16]
sum: 40
```

```
threes: [0, 3, 6, 9, 12, 15, 18]
twos: [2, 4, 8, 10, 14, 16]
sum: 33
```

The code would not compile.

```
threes: [3, 6, 9, 12, 15, 18]
twos: [2, 4, 8, 10, 14, 16]
sum: 40
```

```
threes: [3, 6, 9, 12, 15, 18]
twos: [2, 4, 8, 10, 14, 16]
sum: 33
```

```
threes: [3, 6, 9, 12, 15, 18]
twos: [2, 4, 8, 10, 14, 16, 20]
sum: 40
```

The code would generate a run time exception.

```
threes: [0, 3, 6, 9, 12, 15, 18]
twos: [2, 4, 8, 10, 14, 16, 20]
sum: 40
```

```
threes: [3, 6, 9, 12, 15, 18]
twos: [2, 4, 8, 10, 14, 16, 20]
sum: 33
```

```
threes: [0, 3, 6, 9, 12, 15, 18]
twos: [2, 4, 8, 10, 14, 16, 20]
```

```
sum: 33
```

QUESTION 3

3 points Save Answer

What would be the output when the **`main()`** method is executed?

```
public class Q3 {
    public static void doSomething(int [] x, int[] y) {
        for (int i = 0; i < x.length; i += 2) {
            x[i] = y[i];
        }
        y = x;
    }

    public static void main(String[] args) {
        int [] x = {2, 4, 6, 8};
        int [] y = {0, 1, 2, 3};

        doSomething(x, y);

        System.out.println(x[0] + "," + x[1] + "," + x[2] + "," + x[3] );
        System.out.println(y[0] + "," + y[1] + "," + y[2] + "," + y[3] );
    }
}
```

- ○ `0,4,2,8`
  `0,1,2,3`
- ○ The code would generate a run time exception.
- ○ `2,4,6,8`
  `0,1,2,3`
- ○ The code would not compile.
- ○ `0,1,2,3`
  `0,1,2,3`
- ○ `0,4,2,8`
  `0,4,2,8`

QUESTION 4

3 points Save Answer

What would be the output when the `main()` method is executed?

```
public class Q4 {
    public static boolean paradox(int v) {
        return -v > 0;
    }

    public static void enigma(int[] a) {
        for (int i = 0; i < a.length; i++) {
            if (paradox(a[i])) {
                a[i] = a[i] * 2;
            } else {
                a[i] = a[i] * 5;
            }
        }
    }

    public static void main(String[] args) {
        int[] values = {2, -6, 5, 8, -3, -4};

        enigma(values);
        for (int i = 0; i < values.length; i++) {
            System.out.println("values[" + i + "] = " + values[i]);
        }
    }
}
```

○
```
values[0] = -10
values[1] = -12
values[2] = -25
values[3] = -40
values[4] = -6
values[5] = -8
```

○
```
values[0] = 4
values[1] = -30
values[2] = 10
values[3] = 16
values[4] = -15
values[5] = -20
```

○ The code would generate a run time exception.

○
```
values[0] = 10
values[1] = -12
values[2] = 25
values[3] = 40
values[4] = -6
values[5] = -8
```

○
```
values[0] = 4
values[1] = 30
values[2] = 10
values[3] = 16
values[4] = 15
values[5] = 20
```

```
values[0] = 10
values[1] = 12
values[2] = 25
values[3] = 40
values[4] = 6
values[5] = 8
```

The code would not compile.

QUESTION 5

3 points Save Answer

What would be the output when the `main()` method is executed?

```
public class Q5 {
    public static void printFun(int test) {
        if (test >= 1) {
            System.out.println(test + " ");
            printFun(test / 2);
            System.out.println(test + " ");
        }
    }

    public static void main(String[] args) {
        int test = 12;
        printFun(test);
    }
}
```

- ○ `12 6 3 1`
  `1 3 6 12`
- ○ `12 6 3`
  `3 6 12`
- ○ The code would generate a **`ArithmeticException`** at run time.
- ○ The code would not compile.
- ○
  ```
  12
  6
  3
  1
  1
  3
  6
  12
  ```
- ○ The code would generate a **`StackOverflowException`** at run time.
- ○
  ```
  12
  6
  3
  3
  6
  12
  ```

QUESTION 6

3 points Save Answer

Assume that the following code is all in the same file. What would be the output when the **`main()`** method is executed?

```
abstract class A {
    int x = 3;

    abstract public int op(int a);
}

class B extends A {
    @Override
    public int op(int a) {
        return x * a;
    }
}

class C extends B {
    @Override
    public int op(int a) {
        return super.op(x * a);
    }
}

public class Inherit {
    public static void main(String... args) {
        A a = new C();
        System.out.println(a.op(2));
    }
}
```

- ◯ The code would not compile.
- ◯ 12
- ◯ 18
- ◯ The code would generate a run time exception.
- ◯ 6
- ◯ 24

QUESTION 7

2 points Save Answer

Assume all of the code below is in a single file. What would be the output when the **`main()`** method is executed?

```
enum Planet {
    MERCURY,
    VENUS,
    EARTH,
    MARS,
    JUPITER,
    SATURN,
    URANUS,
    NEPTUNE
}

public class SolarSystem {
    private int number = 0;

    public SolarSystem(Planet planet) {
        switch (planet) {
            case MERCURY:
                number = 1;
            case VENUS:
                number = 2;
            case EARTH:
                number = 3;
                break;
            case MARS:
                number = 4;
            case JUPITER:
                number = 5;
            case SATURN:
                number = 6;
                break;
            case URANUS:
                number = 7;
            case NEPTUNE:
                number = 8;
        }
    }

    public int getNumber() {
        return number;
    }

    public static void main(String[] args) {
        SolarSystem solarSystem = new SolarSystem(Planet.MARS);
        System.out.println(solarSystem.getNumber());
    }
}
```

- ○ The code would not compile.
- ○ 4
- ○ The code would generate a run time exception.
- ○ 6
- ○ 0

QUESTION 8

2 points Save Answer

Select the code fragments that are required to correctly implement a method that can print out the contents of a list of any type of elements.

```
public static void printList(/* 1 */ items) {
    for (/* 2 */ item : items) {
        System.out.println(item);
    }
}
```

From the options below, select the code fragments that are required at positions **`/* 1 */`** and **`/* 2 */`** in the method above.

- ☐ `/* 1 */List<Object>`
- ☐ `/* 2 */Object`
- ☐ `/* 1 */List<T>`
- ☐ `/* 1 */List<T extends Object>`
- ☐ `/* 1 */List<?>`
- ☐ `/* 2 */T`
- ☐ `/* 2 */?`

QUESTION 9

6 points Save Answer

Given the following definition of a class **`Customer`**:

```
import java.util.List;
import java.util.ArrayList;

public class Customer {
    private String name;
    private int id;

    public Customer(String name, int id) {
        this.name = name;
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public int getId() {
        return id;
    }

    @Override
    public String toString() {
        return name + ": " + id;
    }

    public static void main(String[] args) {
        List<Customer> customers = new ArrayList<>();
        customers.add(new Customer("Xue", 1234));
        customers.add(new Customer("Jane", 234));
        customers.add(new Customer("Nina", 1011));
        customers.add(new Customer("Harry", 2145));
    }
}
```

Using a **`forEach`** loop, write a lambda expression to only print out customers with an id between 1000 and 1999 inclusive.

QUESTION 10

8 points Save Answer

The set difference between two sets is defined by the elements that are in the first set but not in the second set. Consider the definition of the method **`setDifference`** below.

```
/**
 * Returns a list of all elements in set x that are not in set y.
 *
 * @require x != null && y != null && !x.contains(null) && !y.contains(null)
 *
 * @ensure \result is a list of all elements in set x but not in set y
 *
 */
public List<Integer> setDifference (List<Integer> x, List<Integer> y) {

}
```

Write the implementation for the method **`setDifference`**.

QUESTION 11

8 points Save Answer

Write a set of black-box test cases for the **setDifference** method defined in the previous question. Do **not** write JUnit tests. Each test case should include the input values, the expected return value or exception thrown, and a brief justification for the test case.

QUESTION 12

8 points Save Answer

Consider the definition of the method **`power`** below:

```
/**
 * @require x >= 0 && y >= 1
 * @ensure \result == x raised to the power y
 */
public static int power(int x, int y) {
    int result = 1;

    while (y != 0) {
        if (y % 2 == 0) {
            x = x * x;
            y = y / 2;
        }

        result *= x;
        y--;
    }
    return result;
}
```

Provide a set of white-box test cases for the method. Do **not** write JUnit tests. Each test case should include the input values, the expected return value or exception thrown, and a brief justification for the test case. Path coverage is expected.

QUESTION 13

12 points Save Answer

Write a **recursive** method **`sumDigits`** that given a non-negative number, returns the sum of all the digits. For example **`sumDigits(5247)`** would return 5 + 2 + 4 + 7 = 18. **Do not** turn the number into a string.

Provide a JavaDoc comment that describes the method and which includes appropriate pre and post conditions that describe its behaviour. This includes the constraints on non-negative numbers.

Hint: recall that modulo 10 will give you the last digit of a number.

QUESTION 14

12 points Save Answer

Assume that the following four classes are all in separate files within the same package.

```
public class A {
    /**
     * @require nums != null && val > 0
     * @ensure \result is the list of all of the
     *         integers n < val that appear in nums
     */
    public List<Integer> getValues(int[] nums, int val) { }
}

public class B extends A {
    /**
     * @require nums != null && nums only contains
     *          positive integers && val > 0
     * @ensure  \result is the list of all of the
     *          integers n < val that appear in nums
     */
    public List<Integer> getValues(int[] nums, int val) { }
}

public class C extends A {
    /**
     * @require nums != null && val > 0
     * @ensure  \result is the list of all of the positive
     *          integers n < val that appear in nums
     */
    public List<Integer> getValues(int[] nums, int val) { }
}

public class D extends A {
    /**
     * @require val > 0
     * @ensure  \result is the list of all of the positive
     *          integers n < val that appear in nums
     */
    public List<Integer> getValues(int[] nums, int val) { }
}
```

The Liskov substitution principle requires that a subtype's overridden methods must satisfy the supertype's method specifications. It guarantees that code expecting the supertype method behaves correctly even though the calls go to the subtype implementation.

1. Does class B satisfy the substitution principle with respect to class A? Explain why or why not.
2. Does class C satisfy the substitution principle with respect to class A? Explain why or why not.
3. Does class D satisfy the substitution principle with respect to class A? Explain why or why not.

QUESTION 15

12 points Save Answer

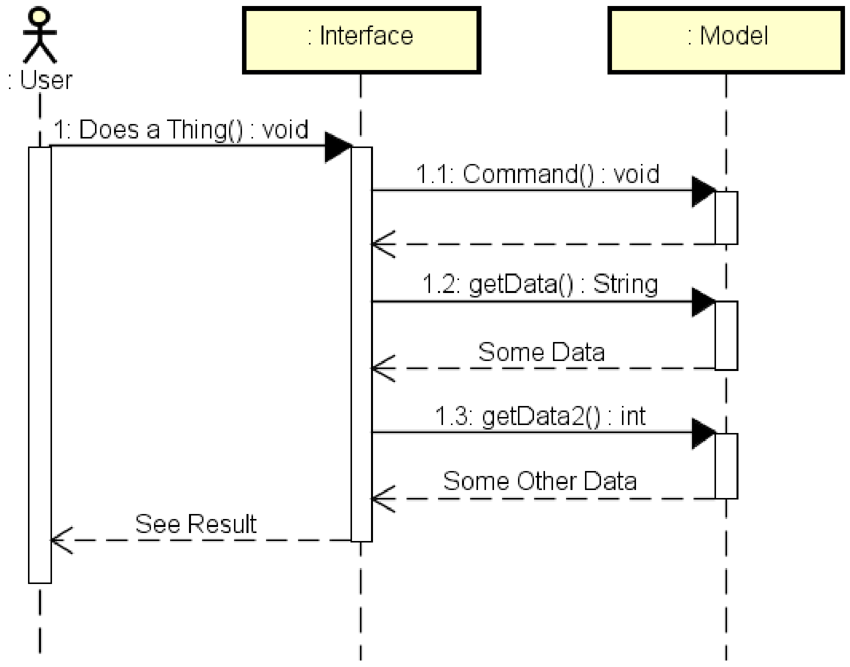
Does the **`Customer`** class below exhibit high or low cohesion? Provide a detailed justification for your answer. Your answer should consider all parts of the class.

```
public class Customer {
    private String name;
    private String streetAddress;
    private String suburb;
    private String postCode;
    private List<Item> order;

    public Customer(String name, String streetAddress, String suburb, String postCode) {
        this.name = name;
        this.streetAddress = streetAddress;
        this.suburb = suburb;
        this.postCode = postCode;
        order = new ArrayList<>();
    }

    public String getName() {
        return name;
    }

    public String getMailingAddress() {
        return streetAddress + suburb + postCode;
    }

    public void addToOrder(Item item) {
        order.add(item);
    }

    public List<Item> getOrder() {
        return order;
    }
}
```

QUESTION 16

12 points Save Answer

When designing a graphical user interface (GUI) it is considered good practice to reduce the coupling between the underlying logical model and the user interface. One way in which this can be achieved is for the user interface to manage all interactions with the model. The following diagram gives an example of this way of managing interactions.



What are the advantages and disadvantages of this approach to designing a GUI? Provide a clear explanation of all the advantages and disadvantages you can identify.

Is this approach commonly used in designing GUIs?