Discussion 4
Prelim 1 review

CS 2110, FA23

Topics

- Procedural programming in Java
- Compile-time and runtime
- Classes
- Testing
- Object-oriented programming
- Exceptions

Procedural programming in Java

Classify the following as either a *primitive type*, a *reference type*, or *not a type name*:

- `Object`
- `char`
- `5`
- `String`
- `null`
- `int[]`

Predict the result of running this program on the given input

```
int[] arr = new int[]{1, 2, 4, 8, 16, 32, 64, 128};
for (int i = 0; i < arr.length; i +=1) {
   int temp = arr[arr.length - i - 1];
    arr[arr.length - i - 1] = arr[i];
    arr[i] = temp;
}
```

Complete this short method given the specification

```
/** Returns a new String with the characters of s in reverse order,
  * ex. reverseString("hello") => "olleh".
  * Requires s is not null.
  * You may not use any Java methods or classes beyond length(),
  * charAt(), and concatenation operators. */

public static String reverseString(String s) {









}
```

Compile-time and run-time

Give an initialization value of w that…?

```
public static void main(String[] args) {
    int x = 8;
    int w;

    try {
        int res = x % w;
        System.out.println(res);

    } catch (RuntimeException re) {
        System.err.println("Whoopsies");
    }
}
```

1. Causes a compile-time error.
    a. In this case do any of our print statements run?
2. Causes an ArithmeticException to be thrown
    a. In this case what gets printed?
3. Causes 0 to be printed.

Determine if the following statements compile:

```
interface I1 {

}

interface I2 {

}

class A implements I2 {

}

class B extends A implements I1, I2 {

}
```

```
B b = new B();
I2 i2 = b;
```

a) `I1 k = (I2) i2;`

b) `I1 k2 = b;`

c) `I1 k3 = i2;`

d) `String s = i2.toString();`

Classes in Java

Class Diagrams

Given the following class, please draw a class diagram:

```
public class Student {
    private String name;
    private String netId;
    private int credits;

    public String name() {
       return name;
    }

    public String netId() {
       return netId;
    }

    public void modifyCredits(int creditChange) {
       credits += creditChange;
    }
}
```

Label the return type, parameters, specification, keywords, types and literals in the method below:

```
 /**
  * This method returns true if every character in String word consists of
  * lowercase english alphabet ('a' - 'z'), and false if otherwise.
  * Requires: word is not null or empty ("").
  */
 public static boolean isAllLowerCase(String word) {
   for (int i = 0; i < word.length(); i++) {
     char currentChar = word.charAt(i);
     if (currentChar < 'a' || currentChar > 'z') {
       return false;
     }
   }
   return true;
 }
}
```

Implement isSolved() according to the specification

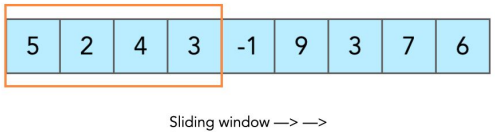
```
/** A class representing a single row of cells in a Sudoku game */
public class SudokuRow {
    /** The values in each of the cells in the row.
     * Each element is either filled with a number 1-9 or is an empty cell, marked by a 0
     * Invariant: Only contains values in the range 0-9 inclusive.
     * Invariant: Each number in range 1-9 inclusive can only appear at most once in the row.
     */
    private int[] cells;

    // Other fields, constructors, and methods omitted

    /** Returns whether the row has been solved. A row has been solved if there are no empty cells in the row
    */
    public boolean isSolved() {
        //TODO
    }
}
```

Testing

Given the method specification, write at least three **black box tests**, stating the input and expected output

Recap: Black box testing is a technique of testing where the functionality of the software is tested by only looking at the specifications and without looking at the code.

The function you are testing is *movingAverage()*. It takes in 2 parameters, an array of integers and a window size (must be a positive integer), and returns an array of doubles representing the average of all integers in the sliding window.



Special cases:

- If the array is empty, return null
- If window size > size of the array, return an array with just one element (average of the list)

Object-oriented programming in Java

What will happen when we try to compile and run A and B?

```
public class Animal {
	public void makeNoise() {
		System.out.println(“This animal is making its call”);
		call();
	}

	public void call() {
		System.out.println(“Grunt”);
	}
}

public class Cat extends Animal {
	public void call() {
		System.out.println(“Meow”);
	}

	public void pet() {
		System.out.println(“Purr”);
	}
}
```

A

```
public static void main(String args[]) {
	Animal oliver = new Cat();
	oliver.makeNoise();
}
```

B

```
public static void main(String args[]) {
	Animal oliver = new Cat();
	oliver.pet();
}
```

Does the following equals() method for the Player class satisfy all the properties of an equivalence relation? If not, which ones does it violate

```
Public class Player{
        public String playerName;
        public int jerseyNo;
        public String team;
        public boolean equals(Object obj) {
                if(!obj instance of Player) {return false;}
                Player pl = (Player) obj;
                if(this.jerseyNo > pl.jerseyNo) {
                   return this.playerName.equals(pl.playerName) && this.team.equals(pl.team);
                }
                return this.playerName.equals(pl.playerName);
        }
}
```

Does Class SuperSonics implement Interface NBATeam? Are there any compile-time errors?

(There are no specifications, so we can't say whether it implements it *correctly*; just interested in whether it compiles for now.)

```
public interface NBATeam {

    public double winPercent();

    public String nextGame();

}
```

```
public class SuperSonics implements NBATeam {
    int gamesPlayed;
    double winPercent;
    String[] schedule;
    public SuperSonics(){
        gamesPlayed = 0;
        this.winPercent = 0.0;
        this.schedule = null;
        //the team no longer exists, so the schedule will always
        be null
    }
    public double winPercent() {
        return winPercent;
    }
    public String nextGame() {
        return schedule[gamesPlayed];
    }
}
```

Exceptions

Exceptions: Try-Catch

- (1) Does this try block throw an exception? If so what exception? (2) What is the final value of the variable b (if the program does not crash)? (3) What is printed out?

```
public class Main {
    public static void main(String[] args) {
        int b = 6;
        try {
            b = 1;
            int a = 3 / 0;
            b = 4;
            System.out.println("one");
        }

        catch (RuntimeException e) {
            b = 3;
        }
    }
}
```

Convert the following method to throw an `Exception` instead of returning -1:

```
public int indexOf(char input) {
    // Iterate over each character in String
    for (int i = 0; i < this.length(); i++) {
        // If current character equals input character
        if (this.charAt(i) == input) {
            return i; // Return the current index
        }
    }

    return -1; // Character not found, return -1
}
```