WIT COMP1000

Final Review

Format

- The exam will be 6-7 problems, with some problems having multiple sub-questions
- You are allowed a single 8.5x11" piece of paper with whatever notes you want on it
  - Can be handwritten or computer printed
  - You may use both the front and back
- No calculators, books, laptops, phones, or anything besides your single page of notes may be used

Format

- Kinds of questions to expect:
  - Explain a program or part of a program
  - Translate between "normal" math expressions and their Java equivalents
  - Write your own code
  - Fix incorrect code / find bugs in code
  - Fill in the blank (in a program)
  - Short answer

Content

- Everything we've covered all semester, including:
  - Input and output
  - Mathematical expressions (order of operations, integer division, etc)
  - `if-else` statements
  - `while`, `do-while`, and `for` loops
  - Methods
  - Arrays
  - File I/O and exceptions
  - Classes

Review Exercises

- The following slides contain exercises that will help you prepare for the exam
- These exercises are all about writing code to help remind you of the things we've done so far this semester
- Refer back to the exam 1 and 2 review slides (and your actual exams) if you need a reminder of the style of questions

Exercise

- Write a program that computes the total cost of buying Cheetos in bulk. The user enters the number of bags purchased, and the program computes and outputs the total cost using these rules:
  - Less than 10 bags purchased, cost is $2/bag
  - Between 10 and 20 bags (inclusive), cost is $1.50/bag
  - More than 20 bags, cost is $1.25/bag

Answer

```
import java.util.InputMismatchException;
import java.util.Scanner;

public class ClassExamples {
	public static void main(String[] args) {
		@SuppressWarnings("resource")
		Scanner keyboardInput = new Scanner(System.in);
		int numItems = 0;
		System.out.print("Enter the number of bags purchased: ");

		try {
			numItems = keyboardInput.nextInt();
		} catch (InputMismatchException ex) {
			System.out.println("Invalid input!");
			System.exit(0);
		}

		double cost;
		if (numItems < 10) {
			cost = numItems * 2;
		} else if (numItems <= 20) {
			cost = numItems * 1.5;
		} else {
			cost = numItems * 1.25;
		}

		System.out.printf("Total cost: $%.2f%n", cost);
	}
}
```

Exercise

- Write a program that asks the user for exactly ten integers and then prints them out in the reverse order given. Use an array to store the values so you can print them out after you have read in all ten.

Answer

```
import java.util.InputMismatchException;
import java.util.Scanner;

public class ClassExamples {
	public static void main(String[] args) {
		@SuppressWarnings("resource")
		Scanner keyboardInput = new Scanner(System.in);

		int[] inputVals = new int[10];
		System.out.printf("Enter exactly %d integers: ", inputVals.length);

		try {
			for (int i = 0; i < inputVals.length; i++) {
				inputVals[i] = keyboardInput.nextInt();
			}
		} catch (InputMismatchException ex) {
			System.out.println("Invalid input!");
			System.exit(0);
		}

		System.out.print("The integers in reverse order: ");
		for (int i = inputVals.length-1; i >= 0; i--) {
			System.out.print(inputVals[i] + " ");
		}
		System.out.println();
	}
}
```

Exercise

- Write a method that is passed a **`double`** array that then squares every value in the array. That is, the method replaces each element in the array with the value of the original element squared. The method must work for any size array. Write a `main()` method to test your method.

Answer

```
public class ClassExamples {
    public static void squareArray(double[] a) {
        for (int i = 0; i < a.length; i++) {
            a[i] = a[i] * a[i];
        }
    }

    public static void main(String[] args) {
        double[] testVals = { 1.1, 2.2, 3.3, 4.4, 5.5 };

        System.out.println("Original values: ");
        for (int i = 0; i < testVals.length; i++) {
            System.out.printf("%.2f%n", testVals[i]);
        }

        squareArray(testVals);

        System.out.println("Squared values: ");
        for (int i = 0; i < testVals.length; i++) {
            System.out.printf("%.2f%n", testVals[i]);
        }
    }
}
```

Exercise

- Write a method that is passed a `String` array and returns the longest `String` (the one with the most characters). If there are multiple `String` objects that are tied for the longest, return the last such `String`. Write a `main()` method to test your method.

Answer

```
public class ClassExamples {

    public static String findLongest(String[] a) {
        String longest = "";
        int maxLength = -1;
        for (int i = 0; i < a.length; i++) {
            if (a[i].length() >= maxLength) {
                maxLength = a[i].length();
                longest = a[i];
            }
        }
        return longest;
    }

    public static void main(String[] args) {
        String[] testVals = { "abcdef", "stuff", "more stuff", "abcdefghij" };

        String answer = findLongest(testVals);

        System.out.println("The longest string was: " + answer);
    }
}
```

Exercise

- Write a program that reads every line of input from the user. The program should count how many total uppercase letters, lowercase letters, and non-letter characters there were across all lines of input. Print out each of the three counts after the end of the input.

Answer

```
import java.util.Scanner;
public class ClassExamples {
	public static void main(String[] args) {
		@SuppressWarnings("resource")
		Scanner keyboardInput = new Scanner(System.in);
		int countLower = 0;
		int countUpper = 0;
		int countOther = 0;

		System.out.println("Enter text here, and use Ctrl-Z to indicate you are finished:");
		while (keyboardInput.hasNextLine()) {
			String line = keyboardInput.nextLine();
			for (int i = 0; i < line.length(); i++) {
				char nextChar = line.charAt(i);
				if (nextChar >= 'a' && nextChar <= 'z') {
					countLower++;
				} else if (nextChar >= 'A' && nextChar <= 'Z') {
					countUpper++;
				} else {
					countOther++;
				}
			}
		}

		System.out.printf("There were %d lowercase letters.%n", countLower);
		System.out.printf("There were %d uppercase letters.%n", countUpper);
		System.out.printf("There were %d non-letters.%n", countOther);
	}
}
```

Exercise

- Write a method named `validateTriangle()` that is given three **`double`** parameters representing the lengths of three sides of a triangle. All three sides should be verified (positive lengths, form a valid triangle). The method must return true if the triangle was valid and return false otherwise. Write a `main()` method to test your method.

Answer

```
public class ClassExamples {
	public static boolean validateTriangle(double a, double b, double c) {
		if (a <= 0 || b <= 0 || c <= 0) {
			return false;
		}
		if ((a >= b+c) || (b >= a+c) || (c >= a+b)) {
			return false;
		}
		return true;
	}
	public static void main(String[] args) {
		if (validateTriangle(1, 1, 1)) {
			System.out.println("1,1,1 is valid!");
		} else {
			System.out.println("1,1,1 is invalid!");
		}
		if (validateTriangle(1, 0, 1)) {
			System.out.println("1,0,1 is valid!");
		} else {
			System.out.println("1,0,1 is invalid!");
		}
		if (validateTriangle(1, 1, 100)) {
			System.out.println("1,1,100 is valid!");
		} else {
			System.out.println("1,1,100 is invalid!");
		}
	}
}
```

Exercise

- Write a program which reads grades from a file named `grades.txt`. Read up to the first 30 grades in the file, compute the average of those grades, and then print out how far away each grade is from the average (the difference of the grade and the average).

Answer

```
import java.io.File;
import java.io.FileNotFoundException;
import java.util.Scanner;

public class ClassExamples {
	public static void main(String[] args) {
		double[] grades = new double[30];
		int count = 0;
		double sum = 0;
		try (Scanner fileInput = new Scanner(new File("grades.txt"))) {
			while (fileInput.hasNextDouble() && count < grades.length) {
				grades[count] = fileInput.nextDouble();
				sum += grades[count];
				count++;
			}
		} catch (FileNotFoundException ex) {
			System.out.println("grades.txt doesn't exist!");
			System.exit(0);
		}

		if (count == 0) {
			System.out.println("No grades in grades.txt!");
			System.exit(0);
		}
		double average = sum / count;
		System.out.printf("Grade differences from the average %.3f:%n", average);
		for (int i = 0; i < count; i++) {
			System.out.printf("Grade %d: %.3f%n", i, grades[i] - average);
		}
	}
}
```

Exercise

- Write a method named `printVals()` that is passed an already opened `PrintWriter` object. Additionally, the method should be passed 4 integer parameters: `start`, `stop`, `increment`, and `printIncrement`. The method must output into the `PrintWriter` object all of the numbers starting at `start`, counting by `increment`, and stopping after `stop`. At most `printIncrement` numbers should be printed in each line of output. Include a `main()` method to test your `printVals()` method.

Answer

```
import java.io.File;
import java.io.FileNotFoundException;
import java.io.PrintWriter;

public class ClassExamples {
	public static void printVals(PrintWriter output, int start, int stop, int increment, int printIncrement) {
		int count = 1;
		for (int i = start; i <= stop; i += increment) {
			if (count % printIncrement == 0) {
				output.println(i);
			} else {
				output.print(i + " ");
			}
			count++;
		}
	}

	public static void main(String[] args) {
		try (PrintWriter fout = new PrintWriter(new File("testOut.txt"))) {
			printVals(fout, 100, 1000, 2, 10);
		}
		catch (FileNotFoundException ex) {
			System.out.println("File testOut.txt not found!");
			System.exit(0);
		}
	}
}
```

Exercise

- Write a program that opens a file named `lines.txt` and prints out the *last* two lines from the file. If there are less than two lines, print an error message.

Answer

```
import java.io.File;
import java.io.FileNotFoundException;
import java.util.NoSuchElementException;
import java.util.Scanner;

public class ClassExamples {
	public static void main(String[] args) {
		try (Scanner fileInput = new Scanner(new File("lines.txt"))) {
			String secondLast = fileInput.nextLine();
			String last = fileInput.nextLine();
			while (fileInput.hasNextLine()) {
				String next = fileInput.nextLine();
				secondLast = last;
				last = next;
			}
			System.out.println(secondLast);
			System.out.println(last);
		} catch (FileNotFoundException ex) {
			System.out.println("lines.txt doesn't exist!");
			System.exit(0);
		} catch (NoSuchElementException ex) {
			System.out.println("lines.txt has less than two lines!");
			System.exit(0);
		}
	}
}
```

Exercise

- Write a class that represents a book. Every book should have a name, number of pages, and year published. Include an appropriate constructor and an `toString()` method. Write a `main()` method in a separate class to test your book class.

Answer

Book.java:

```
public class Book {
	private String title;
	private int pageCount;
	private int yearPublished;

	public Book(String bookTitle, int bookPageCount, int bookYearPublished) {
		title = bookTitle;
		pageCount = bookPageCount;
		yearPublished = bookYearPublished;
	}

	public String toString() {
		String output = "Book: '" + title + "'";
		output += String.format(", Published: %d", yearPublished);
		output += String.format(", Length: %d pages", pageCount);
		return output;
	}
}
```

ClassExamples.java:

```
public class ClassExamples {
	public static void main(String[] args) {
		Book hobbit = new Book("The Hobbit", 310, 1937);
		Book iRobot = new Book("I, Robot", 272, 1950);

		System.out.println(hobbit);
		System.out.println(iRobot);
	}
}
```

Exercise

- Write a class that represents a planet. Every planet has a radius and a distance from the sun. Include an appropriate constructor, a method that lets you set the distance separately, a method that returns the approximate volume of the planet (volume of a sphere is $4\pi r^3/3$), and a `toString()` method. Write a `main()` method in a separate class to test the planet class.

Answer

Planet.java:

```
public class Planet {
	private double radius;
	private double distanceFromSun;
	public Planet(double planetRadius, double planetDistance) {
		radius = planetRadius;
		distanceFromSun = planetDistance;
	}
	public void setDistance(double planetDistance) {
		distanceFromSun = planetDistance;
	}
	public double volume() {
		return (4.0/3.0) * Math.PI * Math.pow(radius, 3);
	}
	public String toString() {
		String output = String.format("Radius: %.0f km, ", radius);
		output += String.format("Distance From Sun: %.0f km, ", distanceFromSun);
		output += String.format("Approximate Volume: %.0f cubic km", volume());
		return output;
	}
}
```

ClassExamples.java:

```
public class ClassExamples {
	public static void main(String[] args) {
		Planet earth = new Planet(6371, 150000000);
		System.out.println(earth);
		earth.setDistance(149597871);
		System.out.println(earth);
	}
}
```

Wrap Up

- We've covered all of the basics of Java programming this semester
- If you are interested in more, Computer Science II covers multi-dimensional arrays, recursion, building graphical interfaces, and much more about classes and inheritance
- We've been using Java, but as you'll see in time most of the basics are the same in other languages and the concepts apply much more broadly than only to Java