You may use one 8.5"x11" page of handwritten notes (dual-sided ok), but no books, computers, calculators, or cell phones during the exam. Please do your own work.

Practice time-management in completing this exam … only 50 minutes to complete exam. (Once solutions are developed, notice how little coding is actually required!)

CIS 200 Practice Exam 3

Name:_____

Note: This is NOT a past exam. This is just a list of practice problems to review the topics that will be on the exam.

1) Write the class `Account`. This class should contain the following:

- An account number, name, and account balance as an **instance variables**
- A **constructor** that takes values for each instance variable as parameters, and initializes each of the instance variables
- A **`deposit` method** that takes a double value as a parameter. This method should add the parameter value to the account balance. You should **throw an appropriate exception** if the parameter value is negative.
- A **`toString` method** that can be used to display the account number, name, and account balance to the screen (contains no print statements)

```
public class Account {
	private int number;
	private String name;
	private double balance;

	public Account(int n, String str, double b) {
		number = n;
		name = str;
		balance = b;
	}

	public void deposit(double add) {
		if (add < 0) {
			throw new IllegalArgumentException();
		}
		balance += add;
	}

	public String toString() {
		Return (“Number: “ + number +
				“\nName: “ + name +
				“\nBalance “ + balance);
	}
} // end class Account
```

2) Create a class called **`TestAccount`** that contains a **`main`** method. Inside the `main` method, create an `Account` object with account number 1234, name "Bob", and balance $451.12. Ask the user to enter a value, and then add that value to the account's balance (by calling the `deposit` method). Finally, display the account information (by calling the `toString` method). Afterward, add how you would account for a possible exception (display error message)

```
import java. util.*;
public class TestAccount {
   public static void main(String[] args) {
         Scanner s = new Scanner(System.in);
         Account a = new Account(1234, "Bob", 451.12);

         System.out.print("Enter deposit amount: ");
         double val = Double.parseDouble(s.nextLine());
         a.deposit(val);          // compiles OK w/o try-catch (verified)
         System.out.println("\nAccount info:\n" + a);
   }
} // end class TestAccount
```

With exception handling …

```
try {   a.deposit(val);}
catch (IllegalArgumentException e) {
         System.out.println("Negative deposits not allowed");   }
```

3) Suppose you have the following variable: **`ArrayList<String> words;`**
And further suppose that ***words*** has been initialized and filled with a bunch of words (Strings).

Write a ***code fragment*** that creates a **HashMap** where the *key* is a length of a word in the ArrayList (int / **Integer**) and the *value* is a count of how many words in the ***words*** list had that length (again int / **Integer**).

After creating the **HashMap**, ask the user to enter a number. Use your **HashMap** to print out how many words in the list have that length (assume value entered by user exist in HashMap).

```
HashMap<Integer,Integer> map = new HashMap<Integer,Integer>();

Iterator<String> it = words.iterator();          iterating through ArrayList

while (it.hasNext()) {
      String cur = it.next();
      if (map.get(cur.length()) == null) {          creates if it doesn't exist
            map.put(cur.length(), 1);
      }
      else {
            int count = map.get(cur.length());     gets current count for that word length
            map.put(cur.length(), count+1);          replaces if current length already exist
      }
}

Scanner s = new Scanner(System.in);
System.out.print("Enter a number: ");
int key = Integer.parseInt(s.nextLine());

int count = map.get(key);          look-up count using the length of the word (key)
System.out.println(count);
```

4) Consider the two classes below that are the start for a game where the user guesses a number and the computer prints "high" and "low" until the user is correct. Read the comments for each method to figure out what they do. Assume that each method does what it says it does, even though you can't see the code.

```
public class Guesser {
    //instance variables are hidden
    /**
     * Guesser sets up a new high/low guessing game
     * @param correct – the number the user is trying to guess
     */
    public Guesser(int correct) {
        //code is hidden
    }

    /**
     * guess lets the user guess a number
     * @param val – the user's guess
     * @return "low" if the guess is too low, "high" if the guess is too big, and
         "equal" 0 if the guess is correct
     */
    public String guess(int val) {
        //code is hidden
    }
}
```

```
public class IO {
    //instance variables are hidden
    /**
     * IO sets up a new IO object
     */
    public IO() {
        //code is hidden
    }
    /**
     * getNum gets a number as input from the user
     * @return A number from the user
     */
    public int getNum() {
        //code is hidden
    }
    /**
     * printResult prints whether a guessed number is bigger than, smaller than, or equal to
     *    another number
     * @param guess The guess
     * @param goal The correct number
     * @param compare Either “low”, “high”, or “equal”
     */
    public void printResult(int guess, int goal, String compare) {
        //code is hidden
    }
}
```

Here, Guesser is the Model and IO is the View. **Write the class GuessingGame, which contains a main method and is the Controller for the game**. Use the Guesser and IO classes (and obey MVC architecture - NO I/O statements) so that your program will do the following when run:

- Randomly generate a number from 1-100 (this is the "correct" number that the user is guessing)
- Repeat the following UNTIL the user correctly guesses the number:
  - Get a guess from the user
  - Print either "high", "low", or "equal" for how the user's guess compares to correct number

```
public class GuessingGame {
    public static void main(String[] args) {
        Random r = new Random();
        int val = r.nextInt(100)+1;
        Guesser g = new Guesser(val);
        IO view = new IO();

        boolean flag = true;
        while (flag) {
            int guess = view.getNum();
            String result = g.guess(guess);
            view.printResult(guess, val, result);
            if (result.equals("equal"))
                flag = false;
        } // end while
    } // end main
} // end class
```

5) Consider the following program:

```
import java.util.*;

public class Test {
   public static void main(String[] args) {
      Scanner s = new Scanner(System.in);

      int[] nums = new int[10];
      for (int i = 0; i < 10; i++) {
         System.out.print("Enter num: ");
         try {
            nums[i] = Integer.parseInt(s.nextLine());
         }// end try
         catch (NumberFormatException nfe) {
            nums[i] = i;
         } // end catch
      }//end for

      for (int i = 0; i < nums.length; i++) {
         System.out.println(nums[i]);
      }//end for
   }//end main
}// end class
```

Suppose the user enters the following data at the prompts when the program runs:

3
4
5.3
7
a
7.2
6
h
bob
3

What will the program print for its output?

3
4
2
7
4
5
6
7
8
3