**SUTD 50.001 Introduction to Information Systems and Programming**

**Problem set 2 Part B**

Note: Please submit your answer to eDimension

**Cohort Questions (Week 4)**

**Session 1**

Title: **TestScores**

Q1. [5 points] Write a program that checks the scores stored in an array. The method, testScores, will throw IllegalArgumentException("Element: " + i + " Score: " + s[i]) if the values in array s are less than 0 or bigger than 100. Use try … catch block in main() to handle this exception.

**Possible answer:**

java.lang.IllegalArgumentException: Element: 3 Score: 102.0

Exception: Values cannot be less than 0 or bigger than 100

Execution continues here

Title: **CoffeeMaker**

Q2. [5 points] You are provided with the following:

public class CoffeeMaker {

private static final int *tooCold* = 55;

private static final int *tooHot* = 85;

public static void makeCoffee(int temperature) throws TooCold, TooHot {

if (temperature <= *tooCold*) throw new TooCold();

if (temperature >= *tooHot*) throw new TooHot();

}

}

class TooCold extends Exception{}

class TooHot extends Exception{}

Implement an exception handler, within makeCoffee, that will catch the thrown user-defined exceptions (if any) and return a String comment as follows:

|  |  |
| --- | --- |
| **Exception** | **String comment** |
| TooCold | “Yuck!” |
| TooHot | “Ouch!” |
| None | “Mmm!” |

**Session 2**

Title: **RecurMulti**

Q3. [5 points] Write a recursive program, RecurMulti that computes the multiplication of two numbers. For example, 7 multiplied by 4 equals 28.

**Test cases**

System.out.println(multiply(4,7));

System.out.println(multiply(0,7));

System.out.println(multiply(4,0));

**Expected output:**

28

0

0

Title: **SumDigits**

Q4. [10 points] Write a recursive method, sumDigits(int number) that will compute the sum of the digits in an integer number. For example, if the number is 55, the return result is 10. If the number is -54, the return result is 9. The method signature is below:

public static int sumDigits(int number)

**Session 3**

Title: **CumulativeSum**

Q5. [10 points] Write a recursive method, cumulativeSum(int data[], int n)

that will compute cumulative sums in an array. To find the cumulative sums, add to each value in the array the sum of the values that precede it in the array. data[] contains the initial values of the array. Note data[] is passed by reference and final results are stored in data[]. You may assume n=1 when called from main().

For example, if the values in the array are [2, 3, 1, 5, 6, 2, 7], the result will be [2, (2) + 3, (2 + 3) + 1, (2 + 3 + 1) + 5, (2 + 3 + 1 + 5) + 6, (2 + 3 + 1 + 5 + 6) + 2, (2 + 3 + 1 + 5 + 6 + 2) + 7] or [2, 5, 6, 11, 17, 19, 26].

Title: **DoubleEachLetter**

Q6. [5 points] Write a recursive method, doubleEachLetter(String s) that will duplicate each character in a string and return the result as a new string. For example, if "book" is the parameter, the result would be "bbooookk". The method signature is below:

public staticString doubleEachLetter(String s)

**Homework Questions (Week 4)**

Title: **ModularProgram**

Q1. [10 points] Write a program that performs modular operation of two integer values, for example x % y. Prompt the user to enter two integer values. Catch an exception thrown if modular by zero. Also, catch all other exceptions using Exception. Your program should be in a loop until not ‘y’ (ignore case) is entered.

**Possible answer:**

Welcome to the modulus computer

Enter two integer values

2

0

Sorry, cannot compute mod by 0

Do another pair of values ? (y)

y

Enter two integer values

1

1.1

Sorry, you must enter two integer values

Do another pair of values ? (y)

N

Title: **Scheduler**

Q2. [10 points] Write a program that allows students to schedule appointments at either 1pm, or 2pm. The scheduled time values must be of integer value either 1 or 2. Use an array of two strings to store the names in the available time slots. Write a loop that iterates as long as the array has a free space. Within a try block, allow the user to enter a time and a name. If the time is free, put the name in the array. If the time is not free, throw a TimeInUseException. If the time is not valid, throw an InvalidTimeException. Use a catch block for each different kind of exception.

After all the time-slots are filled, print the names stored in each time-slot.

Title: **SongCard**

Q3. [15 points] Create a class SongCard that represents a gift card for the purchase of songs online.

It should have the following private attributes:

• songs—the number of songs on the card

• activated—true if the card has been activated

and the following methods:

• SongCard(n)—a constructor for a card with maximum of n songs.

• activate()—activates the gift card.

• buyASong()—records the purchase of one song by decreasing the number of songs left for purchase. Throw an exception if the gift card is either completely used or not active.

• songsRemaining()—returns the number of songs that can be purchased.

In the main method, the following activities are implemented.

(a) Invoke the contructor, SongCard(n) with n=10 songs and set card activate to false.

(b) Try to buy a song. Use try … catch block to catch the exception if card is not

activated. Print an error message.

(c) Activate the card by changing card activate from false to true

(d) Try to buy 11 songs. Use try …catch block to catch the exception if card is completely used. Print an error message.

**Possible solution:**

Card has 10 songs and is not activated

Trying to buy a song

Caught error: Card not activated

Card has 10 songs and is not activated

Activating the card

Card has 10 songs and is activated

Buying songs

Bought a song: Card has 9 songs and is activated

Bought a song: Card has 8 songs and is activated

Bought a song: Card has 7 songs and is activated

Bought a song: Card has 6 songs and is activated

Bought a song: Card has 5 songs and is activated

Bought a song: Card has 4 songs and is activated

Bought a song: Card has 3 songs and is activated

Bought a song: Card has 2 songs and is activated

Bought a song: Card has 1 songs and is activated

Bought a song: Card has 0 songs and is activated

Caught error: No more songs on the card

Card has 0 songs and is activated

Title: **Handshake**

Q4. [10 points] There are n people in a room. Each person shakes hands once with every other person if n is greater than 1. If n is 0 or 1, the total number of handshake is 0. What is the total number of handshakes in the room? For example, if n is 10, the total number of handshakes is 45. If n is 3, the total number of handshakes is 3. Write a recursive method to solve this problem. Handle negative numbers by raising an appropriate exception. The method signature is below:

public static int handshake(int n)

// where handshake(n) returns the total number of handshakes for n people in the room.

**Sample Input:**

System.out.println(handshake(10);

System.out.println(handshake(3));

System.out.println(handshake(-4));

**Output:**

45

3

java.util.InputMismatchException: Sorry. Number of persons cannot be negative.

Title: **SumOdds**

Q5. [15 points] Write a recursive method, sumOdds (int number) that will compute the sum of all the odd digits in an integer number. For example, 10134 the return value is 6. If the number is -54321, the return result is -1. The method signature is below:

public static int sumOdds(int i)

Title: **ReverseLetter**

Q6. [10 points] Write a recursive method, reverse(String s) that will reverse the order of the characters in a given string and return the result as a new string. For example, if "book" is the parameter, the result would be "koob". The method signature is below:

public static String reverse(String s)

Title: **RecursiveStringReverse**

Q7. [10 points] Write a recursive method that reverses the order of the words in a string. No credit will be given for non-recursive code. Assume that all words are delimited by single space, and there is no punctuation. The method signature is below:

public static String recurStringReverse (String s)

**Input:**

man ate fish

**Output:**

fish ate man