جامعة جدة
University of Jeddah

College of Computer Science & Engineering

Department of Computer Science and Artificial Intelligence

CCCS214: Object-Oriented Programming II

Lab 1: Exceptions and I/O Streams

Lab Objectives

- Be able to write code that handles an exception.
- Be able to write code that throws an exception.
- Be able to write a custom exception class.

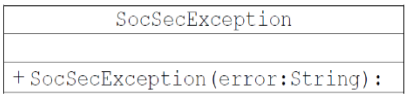
Introduction

This program will ask the user for a person's name and social security number. The program will then check to see if the social security number is valid. An exception will be thrown if an invalid SSN is entered.

You will be creating your own exception class in this program. You will also create a driver program that will use the exception class. Within the driver program, you will include a **`static`** method that throws the exception. Note: Since you are creating all the classes for this lab, there are no Student Files associated with this lab.

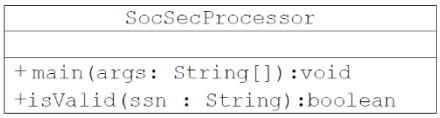
Task #1 Writing a Custom Exception Class

1. Create an exception class called `SocSecException`. The UML diagram for this class is below.



The constructor will call the superclass constructor. It will set the message associated with the exception to *"Invalid social security number"* concatenated with the error string.

2. Create a driver program called `SocSecProcessor`. This program will have a `main` method and a static method called `isValid` that will check if the social security number is valid.



Task #2 Writing Code to Handle an Exception

1. In the `main` method:

   a. The `main` method should read a name and social security number from the user as `String` objects.
   b. The `main` method should contain a `try-catch` statement. This statement tries to check if the social security number is valid by using the method `isValid`. If the social security number is valid, it prints the name and social security number. If a `SocSecException` is thrown, it should catch it and print out the name, social security number entered, and an associated error message indicating why the social security number is invalid.
   c. loop should be used to allow the user to continue until the user indicates that they do not want to continue.

2. The **`static`** `isValid` method:

   a. This method throws a `SocSecException`.
   b. Returns `true` if the social security number is valid, `false` otherwise.
   c. The method checks for the following errors and throws a `SocSecException` with the appropriate message.
      i. Number of characters not equal to 11. (Just check the length of the string)
      ii. Dashes in the wrong spots.
      iii. Any non-digits in the SSN.

   Hint: Use a loop to step through each character of the string, checking fora digit or hyphen in the appropriate spots.

3. Compile, debug, and run your program. Sample output is shown below with user input in bold.

OUTPUT (boldface is user input)

```
Name? Sam Sly
SSN? 333-00-999
Invalid the social security number, wrong number of characters
Continue? y
Name? George Washington
SSN? 123-45-6789
```

```
George Washington 123-45-6789 is valid
Continue? Y

Name? Dudley Doright
SSN? 222-00-999o
Invalid the social security number, contains a character that is
not a digit
Continue? y
Name? Jane Doe
SSN? 333-333-333
Invalid the social security number, dashes at wrong positions
Continue? n
```

Task #3 Binary Files (Reading and Writing)

Based on what you have studied during lectures about binary files Complete the missing codes in the two following classes.

1. WritingBFile.java
2. ReadingBFile.java
3. Notes: these codes are similar to what you have seen in class please name the file (`Num.dat`)

```
public class WritingBFile {

	public static void main(String[] args)
		 throws IOException {
		// An array to write to the file
		int[] numbers = {3, 5, 7, 9, 11, 13, 15};
		// Create the binary output objects.
		// complete here

		System.out.println("Writing the numbers to the file...");

		// Write the array elements to the file.
		for (int i = 0; i < numbers.length; i++) {
			System.out.println("writing: "+numbers[i]);
			// complete
		}
		System.out.println("Done.");
		// Close the file.
		// complete
	}
}
```

```
public class ReadingBFile {

	public static void main(String[] args)
		  throws IOException {
		int number; // A number read from the file
		boolean endOfFile = false; // EOF flag

		// Create the binary file input objects.
		//Complete

		// Read the contents of the file.
		while (!endOfFile) {
		   try { // Complete

		   } catch (EOFException e) {
		       endOfFile = true;
		   }
		}
		System.out.println("\nDone.");
		// Close the file.
		//Complete
	}
}
```