Silver Bullet University

Java Fundamentals

Project 1

You must submit your source code and compiled version for each question and compressed them all in one Archive file

Question 1 – Expected time (1 – 3 hours)

Design a class named **MyInteger**. The class contains:

- An **int** data field named **value** that stores the **int** value represented by this object.
- A no-arg constructor **MyInteger()** that ***calls the next constructor*** to create a MyInteger object of value 0.
- A constructor **MyInteger(int value)** that creates a MyInteger object for the specified **int** value.
- A setter and getter methods that sets/returns the **int** value.
- The methods **isEven()**, **isOdd()**, and **isPrime()** that return **true** if the value in this object is even, odd, or prime, respectively.
- The static methods **isEven(int)**, **isOdd(int)**, and **isPrime(int)** that return **true** if the specified value is even, odd, or prime, respectively.
- The static methods **isEven(MyInteger n)**, **isOdd(MyInteger n)**, and **isPrime(MyInteger n)** that return **true** if the specified object represents an even, odd, or prime integer, respectively.
- The methods **add(MyInteger n)**, **sub(MyInteger n)**, and **mul(MyInteger n), and div(MyInteger n)** that *returns a new **MyInteger** object* that stores the result of adding value of current object to value of parameter **n**, subtracting value of parameter object **n** from value of current object, multiplying value of parameter object **n** by value of current object and dividing value of current object by value of parameter object **n** if it is not zero (returns **null** if parameter is zero), respectively.
- The methods **equals(int n)** and **equals(MyInteger n)** that return **true** if the value in this object is equal to the specified value of integer **n** or object **n**.

Draw the UML diagram for the class and then implement the class. Write a client program that tests all methods in the class by

- Creating two objects of type **MyInteger** with values 5 and 24.
- Trying all methods on these two objects as shown in sample run.
- Trying the static method **isPrime**()on number 15.
- Trying the static method **isOdd**() on number 45.

Name your classes MyInteger and TestMyInteger.

Sample Run:

```
n1 value is 5
n1 is even? false
n1 is prime? true
n2 value is 24
n2 is odd? false
n1 is equal to n2? false
n1 is equal to 5? true
n1 value after n1.add(n2) 29
n2 value after n2.sub(n1) 29
n1 value after n1.mul(n2) 120
n2 value after n2.sub(n1) 19
15 is prime? False
45 is odd? true
```

Question 2 – Expected time (2– 4 hours)

Declare a class **Cafe** with the following:

- Private Attributes ***coffeePrice***, ***teaPrice***, ***donutPrice***, ***coffeeTotQty, teaTotQty, donutTotQty, discount***, ***subtotal, discountedPrice*** and ***total***.
    - ***discount*** is a number between 0-100 and it represents a percentage,
    - ***coffeePrice***, ***teaPrice*** and ***donutPrice*** represent price of items.
    - ***coffeeTotQty***, ***teaTotQty*** and ***donutTotQty*** represent quantities cups and donuts in inventory.
    - ***subtotal*** is bill total before discount.
    - ***discountedPrice*** is the amount discounted.
    - ***total*** is total of final bill

- ***setters*** Methods for the attributes ***coffeePrice***, ***teaPrice***, ***donutPrice***, ***coffeeTotQty, teaTotQty, donutTotQty, discount.*** These methods must validate the data (e.g., price can not be negative)**.**
- ***getters*** Methods for all attributes***.***
- A private method **calculateSubTotal(int coffeeQty, int teaQty, int donutQty)** to calculate the ***subtotal*** of the bill before discount and store it.
- A private method **calculateTotal(int coffeeQty, int teaQty, int donutQty)** to calculate the total cost of the bill, including the discount and store values in ***total*** and ***discoutnedPrice***. It should use method **calculateSubTotal().**
- Public method **double order(int coffeeQty, int teaQty, int donutQty).** This method is called to process a customer's order. It calls the other two methods **calculateTotal()** and **display()** to calculate and display the bill to the user. It receives the number of coffee cups, tea cups and donuts that the user ordered and returns bill total. It should check first if the Café has enough cups and/or donuts before initiating and order. If the Café has inventory shortage then it prints **"Error: no enough cups and/or donuts"** and returns 0.

- Public void method **display(int coffeeQty, int teaQty, int donutQty)** to display an itemized bill as follows: (assume the Cafe sells coffee for SR 5.50 a cup, Tea for SR 3.50 and Donuts for SR 2.25 and discount is 10%)

```
------------------------------------------------------------------------------------------
Item                                Quantity                        Price
------------------------------------------------------------------------------------------
Coffee                                 3                           SR 16.50
Tea                                    2                           SR 7.00
Donuts                                 2                           SR 4.50
------------------------------------------------------------------------------------------
Sub total                                                          SR 28.00
Discount                             (%10)                         SR 2.80
------------------------------------------------------------------------------------------
Total                                                              SR 25.20
```

Do the following:

(1) Declare the class **Cafe** in a separate file called **Cafe.java**.

(2) Write the main program to test class **Cafe** using Class **TestCafe.java.** First, Create an object **c1** or type **Cafe** to represent a Cafe that sells coffee for SR 5.50 a cup, Tea for SR 3.50 and Donuts for SR 2.25. It has 100 cups for tea, 100 cups for coffee and 50 donuts. Then, you should read, calculate and display bills for several customers using a menu driven program (*Hint*: use do while loop for the menu as done in lab06). Your program should display a menu with 2 options:

> 1) Order: to order a drink or/and a donut to a customer.
> 2) Quit: to end program
> Then it should read the option as an integer number.

(3) When the user enters 1 for Order, your program should Read the order, calculate and display bill for customer. Use method ***order*** of object ***c1*** to do all this. When the user enters 2 for Quit, print the total sales for all the operations.

Sample Run

```
**********************************************************************
*                       Welcome to Cafe :)                           *
*                       ---------------------------                  *
*       Please enter one of the following options:                   *
*       1) order ==> this allows you to order a drink                *
*       2) quit ==> to end this program                              *
*                                                                    *
**********************************************************************
Enter your option :> 1
Please, enter order (#cups of coffee, #cups of tea and #donuts: 3 2 2


---------------------------------------------------------------------------------
Item                     Quantity                  Price
---------------------------------------------------------------------------------
Coffee                   3                         16.5
Tea                      2                         7.0
Donuts                   2                         4.5
---------------------------------------------------------------------------------
Subtotal                                           28.0
discount                 (%10)                     2.8000000000000003
----------------------------------------------------------------------------------
Total                                              30.8
----------------------------------------------------------------------------------


**********************************************************************
*                       Welcome to Cafe :)                           *
*                       ---------------------------                  *
*       Please enter one of the following options:                   *
*       1) order ==> this allows you to order a drink                *
*       2) quit ==> to end this program                              *
*                                                                    *
**********************************************************************
Enter your option :> 2
Total sales = 30
Thanks. Goodbye!
```

Question 2 – Expected time (3 – 8 hours)

Problem Description

You need to write a program for a system to manage the books in a librar y. Your system should be able to add books to a library archive, retrieve a book by ISBN, delete a book given ISBN and return the total number of books that have the same author.

The program should enable a library clerk to complete the following tasks

- Add a book to the collection of books in the library.
- Delete a book from the collection of books in the library.
- Find the information about a book given its ISBN.
- Given author name, return the total number of books for that author.
- Print all the books.
- Print all the books with the same *genre* (type of book).

Assumptions:

- Book ISBN is unique; no two books can have the same ISBN.
- ISBN must be checked and validated using a specific formula before adding the book to the archive (see method verifyISBN)
- When adding a book, a reference code is generated to make the classifying procedure easier. A book reference code for the librar y is taken from the
  book title and the author name (see method generateReference).
- There is a counter for the number of books (see numOfBooks) that will be incremented whenever a book is added successfully and decremented when a book is deleted successfully.

Sample Run

```
*********************************************************************
*                                   Welcome to Silver Bullet Library :)
*                                   -------------------------------------------
*          Please enter one of the following options:
*          1) Add a book
*          2) Delete a book
*          3) Find a book
*           4) List all books
*          5) List books for a given genre
*          6) Number of books for a given author
*          7) Total number of books
*          8) Exit
*
*********************************************************************

Enter your option :> 1
Please, enter the book details #ISBN, author, title, and genre.0200 Ali
Java programming
The book has been added.

*********************************************************************
*                                   Welcome to Silver Bullet Library :)
*                                   -------------------------------------------
*          Please enter one of the following options:
*          1) Add a book
*          2) Delete a book
*          3) Find a book
*           4) List all books
*          5) List books for a given genre
*          6) Number of books for a given author
*          7) Total number of books
*          8) Exit
*
*********************************************************************

Enter your option :> 4
ISBN: 200
Author: Ali
Title: Java
genre: programming

*********************************************************************
*                                   Welcome to Silver Bullet Library :)
*                                   -------------------------------------------
*          Please enter one of the following options:
*          1) Add a book
*          2) Delete a book
*          3) Find a book
*           4) List all books
*          5) List books for a given genre
*          6) Number of books for a given author
*           7) Total number of books
*           8) Exit
*
*****************************************************************
Enter your option :> 8
Thanks. Goodbye!
```

Book Class

This class represents a book and its name is Book. It holds book's information.

Here are the **attributes** of the class:

- An **int** data field named **ISBN** hat holds ISBN number. Each ISBN is 4-digit integer that represents the International Standard Book Number.
- A String data field named **author** that holds author name (assume each book has a single author).
- A String data field named **title** that holds book's title.
- A String data field named **genre** that holds book's genre (type of book such as classic, romance, fiction, nonfiction, ...etc).
- A String data field **refCode** that stores the book reference for the library. The book reference is taken from the book title and author name and generated using method **generateReference**() (see methods below).

The methods are defined as follows (all public):

- **Book**: A default constructor.
- **Book (ISBN, author, title, genre)**: constructor that initializes new book with the initial values from the user.
- **Setter methods** (one for each): That sets the values for: (ISBN, author, title, genre).
- **Getter Methods** (one for each): That returns the values of: ISBN, author, title, genre, and refCode.
- A method named **generateReference()** that generates and store a reference for the book in refCode. The reference is of type *String* and it is formed by taking the first two characters of the author name and the first two characters of the book genre and separates them with a dash.
  Example: author = Doyle, genre = Novels → reference code = DO-NO
  <u>**Hint**</u>: Use method **charAt(i)** of class String to get a character at a certain index i in a String (index starts from 0).
  For Example: if value of a String variable **s** is "abc" then **s.charAt(2)** will return 'c'.
- A method named **verifyISBN(int ISBN).** Given an ISBN, it returns true if the entered ISBN is correct and false otherwise. The ISBN is a 4 digit integer where the fourth digit is the *control digit* that checks if the ISBN is correct.

How to verify an ISBN?

Given ISBN = $n_1n_2n_3n_4$ the formula for checking correctness is as follows:

$$(n_1 \times 3 + n_2 \times 2 + n_3 \times 1) \bmod 4 = n_4$$

In other words: the result of this formula must be equal to the *control digit*.

Example: ISBN = 0200 is correct, while 1234 is not correct (use the formula and check!)

Hint: to get each single digit in a number, use similar idea to the on e used in previous assignment

Hint: to get each single digit in a number, use similar idea to the one used in previous assignment .

- A method named **printBookInfo()** that prints a description for the book. It uses the following format:

```
Title: title
Author: author
ISBN: ISBN - Reference Code : referenceCode
Genre: genre
```

- A method named **equals(Book b)** given another Book object, it check s if the two objects (the calling object and b) are equal or not. Hint: the Book ISBN is unique; no two books can have the same ISBN.

Draw the UML diagram for class Book **then implement the class.**

Library Class

This class represents the Library. Class Library contains one array of objects that holds all the books information.

Here are the **attributes** of the class:

- **Book[] libraryBooks:** an array of objects of type Book which holds the list of books in the library.
- **numOfBooks:** stores number of books currently in the library
- **MAX_SIZE:** a public static final attribute that stores the maximum number of books that the library can handle

The **methods** are defined as follows (all public):

- **Library**: a default constructor that creates the array and sets numOfBooks to zero.
- addBook: Adds a book to the collection of books given its ISBN, author, title and genre (follow this order when implementing your

method). The new book is added to the end of the list. This method *returns true* if the add operation was completed successfully, and *false* otherwise. The book is successfully added if its ISBN is correct (**Hint**: use method verifyISBN) and the book is not al ready added before (**Hint:** use the method findBook).
(**Note**: to make things easy for you, first implement this method with out these checks, then add the checks gradually. *You will get a partial grade* if your method does not do the checks).

- addBook: Adds a book to the collection of books given **a Book object**. The new book is added to the end of the list. This method *returns true* if the add operation was completed successfully, and *false* otherwise.
- deleteBook: given an ISBN, the book with the given ISBN is deleted from the library. You should delete a book by copying the last book in the list in place of the deleted book.
- findBook: given a book's ISBN, returns the index of the book in the library if the book is found, otherwise it returns -1.
- findBook: given **a Book object**, returns the index of the book in the library if the book is found, otherwise it returns -1.
- printAll: prints all the books in the library. Nothing will be printed if there are no books. This method should use the method printBookInfo.
- printGenre: given a book genre *g*, prints all books in the library that belongs to the same genre *g*. Nothing will be printed if there are no books for that genre. This method should use the method printBookInfo.
- getNumberOfBooksByAuthor: given an author name, returns the total number of books by the same Author.
- getNumberOfBooks: returns the total number of books in the library.
- getLibraryBooks: returns libraryBooks.
- setNumOfBooks: sets the value of numOfBooks.

The code is provided for the last two methods:

```
public Book[] getLibraryBooks() {
      return libraryBooks;
}
public void setNumOfBooks(int n) {
      numOfBooks = n;
}
```

Draw the UML diagram for class Library **then implement the class.**

Main Class

The main class is class TestLibrary which is the class
that you are going to use to test your program. It contains main method, which presents a menu for the user asking him what he would like to do, as follows:

1) Add a book
2) Delete a book
3) Find a book
4) List all books
5) List books for a given genre
6) Number of books for a given author
7) Total number of books.
8) Exit

Question 3 – Expected time (5 – 10 hours)

Problem Description

Write a program to manage a Hangman game. The game randomly generates a word and prompts the user to guess one letter at a time, as shown in the sample run below. Each letter in the word is displayed as an asterisk *. When the user makes a correct guess, the actual letter is then displayed. When the user finishes a word, display the number of misses and ask the user whether to continue to play with another word.

Sample Run

```
Welcome to Hangman game. Are you ready? OK, let us start:
(Guess) Enter a letter in word ******* > p
(Guess) Enter a letter in word p****** > r
(Guess) Enter a letter in word pr**r** > p
        p is already in the word
(Guess) Enter a letter in word pr**r** > o
(Guess) Enter a letter in word pro*r** > g
(Guess) Enter a letter in word progr** > n
        n is not in the word
(Guess) Enter a letter in word progr** > m
(Guess) Enter a letter in word progr*m > a
The word is program. You missed 1 time
Do you want to guess another word? Enter y or n> n
Goodbye!
```

Hangman Class

Your class Hangman has the following **attributes**:

- A String[] array words to store words of the game, as follows:
  ```
  // Add any words you wish in this array
  String[] words = {"program", "java", "csc111", …};
  ```
- A char[] array hiddenWord to store the current word you want the user to guess
- A char[] array guessedWord to store the user guesses. guessedWord is always filled up with * at the beginning and then its content (characters) change as user guesses them.
- A public Scanner object input to read the user's input.

Your class has the following **methods**:

- A constructor method that creates the arrays hiddenWord and guessedWord.

- A helper (private) method indexOf that receives a character c then it searches for c in hiddenWord. If c is found then its index i is returned, otherwise method returns -1.
- A helper (private) method setCharAt that takes an index i, a character c and a char array arr as parameters. The method then stores the character c in array arr at specified index i.
- A helper (private) method pickWord that generates a random index and then picks and returns a word from the array words to start a new round of the game. Use method nextInt(int bound) from class java.util.Random to generate a random index less than the length of the array words. The method returns a random int value between 0 (inclusive) and the specified value bound (exclusive).
- A helper (private) method copyStringToArray that:
  1. Receives a String s
  2. Returns a new char array using the String method s.toCharArray().
- A helper (private) method printWord that is used to print the array guessedWord.
- A helper (private) method isComplete that checks if the guessedWord still have * in it or not. If there are no * then it returns true (indicating that all letters have been guessed) otherwise it returns false.
- A helper (private) method playOneRound that let the user plays one round of the game (does not receive or return any values).
  Here is how this method (**i.e. core of your program**) works:
  1. It starts by picking a random word from the array words.
  2. Then it initializes the array hiddenWord with the picked up string using copyStringToArray.

  3. After that it initializes each character in array guessedWord to a *.
  4. Then it starts a loop. In this loop:
     - i. It checks if all letters are guessed correctly using method isComplete. If so then it prints the word, using printWord, prints number of misses and terminates. Otherwise, loop continues.
     - ii. It asks the user to enter a character.
     - iii. If the character is not found then it increments number of misses.
     - iv. Otherwise, it starts a nested loop in which:
       1. It uses indexOf to check if the character is in the hiddenWord.
       2. If it is found, then it uses method setCharAt to change the corresponding character in array guessedWord from * to the character itself.
       3. Then it uses same method to change character in hiddenWord to $ (so that we do not find it again).
       4. Then it repeats this nested loop again.

v. Then it repeats the whole loop again.

- A public method play where the whole game goes on. This method does not receive or return any values. It only asks the user if he wants to play a new round of the game. If the user wants to do so then it calls playOneRound otherwise it exits.
- A public method getWords that returns the array words.
- A public method getHiddenWord that returns the array hiddenWord.

The code is provided for the last two methods:

```
public String[] getWords() {
        return words;
}
public char[] getHiddenWord() {
        return hiddenWord;
}
```

Draw the UML diagram for class Hangman **then implement the class.**

Main Class

Write a class TestHangman with main function the simply creates an object of type Hangman and then calls method play.