Unit 3 Programming Problems Worksheet

Programming Problem 1 – Room Class

Create a class that describes one room in a house. The room will have walls that are painted a color. The floor will either be hard wood, carpeting, or tiled. The room may or may not have windows, but it is possible to have multiple windows in the room. This class must include a constructor that takes no input arguments and also an overloaded constructor that takes an input argument for each attribute. The class must include a set and get method for each attribute. The class must also include a method that will print out the current values of all attributes to the console. Create a main method that is in a separate class. The main method will create one room that is painted yellow with hard wood flooring, and 1 window. The main method must create a second room that is painted purple, has tiled floors, and has no windows. The main method must also create a room that is carpeted and painted white with 3 windows. Each of these rooms must be created and printed out in the main method.

Grading Rubric

Task	Points
Attributes	1
Constructors	1
Set methods	1
Get methods	1
Print method	1
Main method for testing and console output	1
Total	**6**

Screenshots

```
Room 1: yellow room with a hard wood floor and 1 windows
Room 2: pruple room with a tiled floor and 0 windows
Room 3: white room with a carpeted floor and 3 windows
```

Programming Problem 2 – High School Locker

A locker for a high school student has a locker number, a student name assigned to it, a number of books stored inside the locker, and also a combination lock. The lock is constructed with a combination that has 3 numbers between 0 and 39. The lock is opened if the user turns the lock to the right to the first combination number, and then left to the second combination number, and then right to third combination number. When the dial is reset it will then be pointing to 0.

Create a class to represent the locker and another class to represent the combination lock. The Locker class will include an attribute that is of type CombinationLock. Each class must include a constructor with no input argument and also a constructor that requires input arguments for all

attributes. Each class must include the appropriate set and get methods. Each class must include a method to print out all attributes.

The CombinationLock class must include the following problem specific methods:

- A method named resetDial that resets the dial to position 0.
- A method named turnLeft that will turn the dial to the left a certain number of ticks from the current position. The number of ticks to turn will be passed as an input argument to this method.
- A method named turnRight that will turn the dial to the right a certain number of ticks from the current position. The number of ticks to turn will be passed as an input argument to this method.
- A method named openLock that will try to open the lock with a combination. The combination that is tried will be passed as three input arguments to this method. This method will return true if the lock opens successfully and will return false if the lock is not opened successfully.

The Locker class must include the following problem specific methods:

- A method named putBookInLocker that puts one additional book into the locker. This method does not take any input arguments and does not return a value.
- A method named removeBookFromLocker that tries to remove one book from the locker. This method does not take an input argument, but does return a value. The book returns true if the book is removed successfully and returns false if the book is not removed successfully.
- A method named openLocker that will try to open the locker. This method will prompt the user to input the three numbers in the combination to try. This method will print out the success or failure of opening the locker.

The main method will be in a separate class from the Locker and the CombinationLock classes. This main method must complete the following:

- Creates locker number 100 for Mickey Mouse. The combination for this locker is 28, 17, 39. This locker will include 3 books.
- Create locker number 275 for Donald Duck. The combination for this locker is 35, 16, 27. This locker will include 0 books.
- Try to open the locker for Mickey Mouse using 15, 18, 18.
- Add three books to Mickey Mouse’s locker.

- Remove one book from Donald Duck's locker.
- After all of these actions have been completed, print out the current state of both lockers to the console.

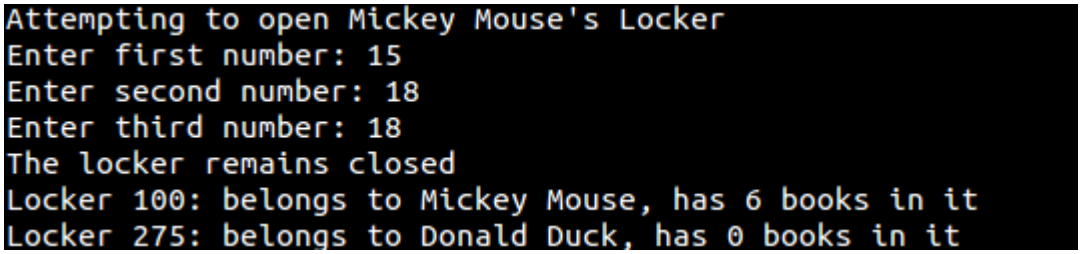
Directions

- You may only use statements that are discussed in the book through Chapter 8.
- Console input and output must be used to solve this problem.

Grading Rubric

Task	Points
CombinationLock - attributes	1
CombinationLock - constructor	1
CombinationLock - resetDial	1
CombinationLock - turnLeft	1
CombinationLock - turnRight	1
CombinationLock - openLock	1
Locker - attributes	1
Locker - constructor	1
Locker - putBookInLocker	1
Locker - removeBookFromLocker	1
Locker - openLocker	1
Main method	2
Console output	1
Total	**14**

Screenshots



Programming Problem 3 - Cycle

[A] Create a class called "Cycle" which has two instance integer variables as properties, "numberOfWheels" and "weight." Create a constructor with two parameters, using the same variable names in the parameter list. Assign each variable to numberOfWheels" and "weight" respectively. Write a separate application to test the class and display its properties. Note: Do

not change the names of the instance variables or the variables listed in the constructor's parameter list.

[B] Edit your class Cycle by adding a default constructor which will assign the default values of 100 to represent the numberOfWheels, and 1000 to represent the weight, by invoking a call to the other constructor. Modify your application created in [A] to test the class.

Directions

[A] Create a class called Cycle

- Declare integer instance variables numberOfWheels and weight as private.
- Include a toString() method in the Cycle class. No set or get methods are included.
- Create a constructor with two parameters, using the same variable names numberOfWheels and weight in the parameter list. Complete the constructor as necessary.
- Create a separate application to test the class.
    - Create an object of the Cycle class.
    - Display the properties of the object.

[B] Add a default constructor

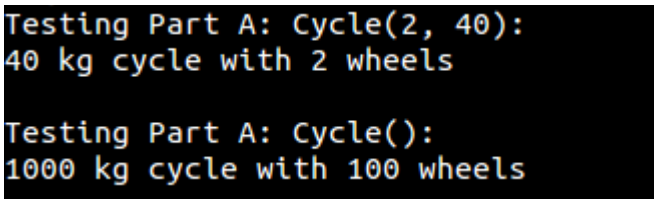
- Edit the default constructor such that the default constructor will invoke the existing constructor with the default values of 100 to represent the numberOfWheels, and 1000 to represent the weight. Invoke the constructor using the "this" reference.
    - Create the second constructor which will receive the two arguments.
- Create a separate application to test the class.
    - Create an object of the Cycle class.
    - Display the properties of the object.

Grading Rubric

Task	Points
Class created and named	2
Constructor(s) completed correctly	2
toString() methods written correctly	1

Successful invocation (constructor to constructor)	2
Properties output correctly	1
Proper documentation / Pseudocode	1
Program works effectively	1

Screenshots



Programming Problem 4 - Book

A **Book** has such properties as **title**, **author**, and **numberOfPages**. A **Volume** will have properties such as **volumeName**, **numberOfBooks**, and an array of book objects (**Book [ ]**). You are required to develop the Book and Volume classes, then write an application (**DemoVolume**) to test your classes. The directions below will give assistance.

Create a class called Book with the following properties using appropriate data types: Title, Author, numberOfPages, Create a second class called Volume with the following properties using appropriate data types: volumeName, numberOfBooks and Book [ ]. The Book [ ] contains an array of book objects.

Directions

- Create a class called Book with the following properties using appropriate data types: Title, Author, numberOfPages,
    - The only methods necessary in the Book class, for this exercise, are the constructor and a toString().
- Create a second class called Volume with the following properties using appropriate data types: **volumeName**, **numberOfBooks** and **Book [ ]**. Book [ ] will contain an array of book objects.
    - The only methods necessary in the Volume class, for this exercise, are the constructor, toString() and getBookArray(). The getBookArray returns a string of book properties for each book.
- Create an application called DemoVolume.

- In the main method,
  - Create an array of book objects to be added to the volume.
  - Create a volume object called volume1.
  - Display the properties of volume1.

Grading

Task	Points
Book class properties created	2
Volume class created correctly	1
Constructors created correctly	2
"this" reference used effectively	1
Array of book objects created	2
Proper documentation / Pseudocode	1
Program works effectively	1
Total	**10**

Screenshots

```
Alice's Adventures in Wonderland:
  Alice in Wonderland by Lewis Carroll(92 pages)
  Through the Looking-Glass by Lewis Carroll(228 pages)
```

Programming Problem 5 – Builder

Write a program which will accept a string from the keyboard, “Java is fun!” Your program must determine the capacity of the string. Then, append a second string, “I love it!” to the first string. At this point you will enter a string “Yes, “ and insert this string so that the output of the entire string will display “Java is fun. Yes, I love it!”

Directions

- Create a class called Builder.
- Using the StringBuilder class, write a program which will accept a string from the keyboard, “Java is fun!”
- Your program must determine and display the capacity of the string.
- Append a second string, “I love it!” to the first string.
- Require the user to enter a string “Yes, “
- Insert the string, “Yes ” at the correct location, such that the output of the entire string will be “Java is fun. Yes, I love it!”
- Display your output using clear and appropriate messages.

Grading

Task	Points
iClass created and named	1
String variables declared and initialized	2
Appropriate use of StringBuilder	2
Appropriate use of string methods	3
Appropriate messages displayed	1
Proper documentation / Pseudocode	1
Total	**10**

Screenshots

```
Please enter the string "Java is fun. ": Java is fun.
Please enter the string "Yes, ": Yes,
The end result is:
Java is fun. Yes, I Love it!
```

Programming Problem 6 – CharacterArray

Write a program which will store the following string into a character array:

6901 Sudley Road Manassas VA

Your program will display each original character, determine and display whether the character is a digit or a letter, and redisplay the character according to the directions below.

Directions

- Create a class called CharacterArray.
- The scanner class is not required
- Write a program which will accept an alpha numeric string at the keyboard.
- Store the string, **of any length**, into a character array, for example:
  NVCC Manassas Campus 6901 Sudley Road Manassas VA 20109
- For each visible character:
  - Display the original character
  - Determine and display whether the character is a digit or a letter
  - If the character is a lower case letter, redisplay the character in upper case
  - If the character is a upper case letter, redisplay the character in lower case
  - If the character is a digit, display it as an * (asterisk)

- Display your output using clear and appropriate messages.

Grading

Task	Points
Class created and named	1
String variables declared and initialized	2
Original characters displayed	1
Appropriate use of string methods	2
Appropriate condition statements used	2
Appropriate messages displayed	1
Proper documentation / Pseudocode	1
Total	**10**

Screenshots

```
Enter an alpha numeric string: NVCC Manassas Campus 6901 Sudley Road Manassas VA 20109
Processed string: NVCC MANASSAS CAMPUS **** SUDLEY ROAD MANASSAS VA *****
```