BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI
CS F213
LAB-2 [Class Design Basics - Part-1]

DATE: 27/08/2018 **TIME: 02 Hours**

1. How to read user inputs in a typical java application?

1.1. Use BufferedReaderclass (See Example1.java and Exercise1.java)

1.2. Use Scanner class (See Example2.java and Exercise2.java)

2. How to design a simple class with proper Attribute(s) and Method(s).

Blue colored lines are to be added to your code for reading user input from the keyboard they will be explained later, for the purpose of this lab just remember to add these lines.

1. How to read user inputs in a typical java application?

There are two ways how you can read user inputs from the keyboard (referred by **stdin** in C and **System.in** in Java).First method is by using the instance of BufferedReader class and the second method is by using the instance of Scanner class.

1.1 **Using BufferedReader for text based user input from the keyboard-** Type the following code in **Example1.java** and save it in D:\Lab2.

```
import java.io.*;    // java.io package is imported
                     // for using BufferedReaderclass
class Example1{

public static void main(String args[])throws IOException{
     /* instantiate InputStreamReader classand pass
          System.in to its constructor */
     InputStreamReader isr =
                    new InputStreamReader(System.in);
     /* instantiate Buffered Reader class and pass the
     reference variable isr which id of type Input Stream
     Reader created in the previous line to the
     constructor of Buffered Reader */

     BufferedReader br= new BufferedReader(isr);

     System.out.println("Enter Your First Name: ");

     /* call readLine method on br reference variable which
     is of type BufferedReader */

     String name = br.readLine();
```

```
        System.out.println("Your name is: " + name);
} // End of main
}// End of class Example1
```

Compile the above code. [Note down what happens during compilation].

Exercise1: Write a program in java to take 10 integer numbers as user input using the BufferedReader and print the sum of these numbers [name the file as Exercise1.java].

[CAREFUL: when reading input with BufferedReader, the input is always read as String, therefore you are required to parse the input to its correct type. In this exercise, use Integer.parseInt() method]

1.2 **Using the Scanner class for text based user input from the keyboard -** Type the following code in **Example2.java** and save it in D:\Lab2.

```
import java.util.Scanner;
class Example2 {
        public static void main(String args[]){
//variable declaration
        int num1;
        double double1;
        String numStr1, numStr2;

        /* instantiate scanner class by passing System.in
        to its constructor */
        Scanner in = new Scanner(System.in);

        System.out.println("Enter an integer: ");
        num1 = in.nextInt(); //reads an int from keyboard
        System.out.println("You entered: " + num1);

        System.out.println("Enter a double: ");
        double1=in.nextDouble();//reads an int from keyboard
        System.out.println("You entered: " + double1);

        System.out.println("Enter your first name ");
        numStr1 = in.next();
        System.out.println("Your name is " + numStr1);

        System.out.println("Enter your surname");
        numStr2 = in.nextLine();
        System.out.println("Your surname is " + numStr2);

        }// End of main() method
}// End of class Example2
```

Compile and run the above code and observe the output?

Exercise2: Write the program description given in Exercise1 in java using the Scanner class [**name the file as Exercise2.java]**

2. Class Design Basics

2.1 Write java implementation for a class named '*Item*' which encapsulates the details of items to be purchased by the customer of the XYZ shop. The class Item is described as follows:

Attributes description:

a) **itemName:String** [Name of the ordered Item of the Customer]
b) **itemidNo:String** [unique identification number of the ordered Item of the Customer]
c) **itemQuantity:int** [quantity of the ordered Item of the Customer]
d) **itemPrice:double** [price of the ordered Item of the Customer]

Methods description:

The class supplies the methods(s) as per the following specification:

a) Any Item instance can be created either by supplying the value for all the instance fields in the order of ***itemName, itemidNo, itemQuantity and itemPrice*** **OR** by supplying the value for ***itemName, itemidNo and itemQuantity*** fields only **OR** by supplying the value for ***itemName, and itemidNo***fieldsonly . If an Item instance is created by providing the value for ***itemName, itemidNo and itemQuantity*** fields only then value for ***itemPrice*** is by default initialized to 500. If an Item instance is created by providing the value for ***itemName and itemidNo*** fields only then value for ***itemPrice*** is by default initialized to 500 and value for ***itemQuantity*** is by default initialized to 1.
b) ***Accessor and mutator methods*** are provided for every instance field.
c) All instance field(s) have a private visibility and all methods have a public visibility.

2.2 Write the java implementation for a class named 'Customer' which encapsulates the details of registered customers of the XYZ shop who buy *Items* (class is described above in 2.1) online. The class Customer is described as follows:

Attributes description:

a) ***name:String*** [Name of the Customer]
b) **idNo:String** [unique identification number of the Customer]
c) **balance:double** [balance amount of the Customer]
d) **item: Item** [item purchased by the Customer]

Methods description:

The class supplies the methods(s) as per the following specification:

a) Any Customer instance can be created either by supplying the value for all the instance fields in the order of ***name, idNo and balance*** **OR** by supplying the value for ***name***

and ***idNo*** fields only. If a Customer instance is created by providing the value for ***name*** and ***idNo*** fields only then value for ***balance*** is by default initialized to 5000.

b) Accessor methods(s) are provided for every instance field.
c) Mutator methods(s) are provided for the instance field ***name*** and ***idNo.***
d) Method ***print()*** to print the details of item (***itemName, itemidNo, iemQuantity and itemPrice*****)** purchased by the customer and balance amount of the customer after purchasing the item.
e) A method named ***buyItem(Item item)*** is supplied. This operation displays details of the item bought and the current balance, if the customer has sufficient balance and the quantity specified by the user is greater than or equal to one. If the customer don't have the sufficient balance, this method displays the message "Insufficient balance" .If the quantity specified by the user is less than one, this method displays the message "Order is not valid".
f) All the fields have private visibility and all methods have a public visibility.

2.3 Write a Test class named TestStore.java which

a) Creates one Customer instance. [You can assume any value for ***name, idNo*** and ***balance*** fields]
b) Creates any two Item instances. [You can assume any value for ***ItemName, ItemidNo, ItemQuantity and ItemPrice*** fields]
c) Display the details of the Items ordered by the customer and customer balance after order.

Note: Assume only one item (any number of quantities) can be ordered at a time by the customer.

Exercises 3.1: Write the java implementation for a class named '*TaxOnSalary*' to calculate tax on salary. The class *TaxOnSalary* is described as follows:

Attributes:

(i) **salary: double // salary to calculate tax**
(ii) **isPANsubmitted:boolean // PAN submission status**

Methods:

The class supplies the operation(s) as per the following specification:

(i) A ***TaxOnSalary*** instance can be created either by supplying the value for the instance field ***isPANsubmitted*** **OR** without supplying value for any field. If ***TaxOnSalary*** instance is created by providing the value for ***isPANsubmitted*** then the value for salary is initialized with 1000.00however it can be reinitialized through the method ***inputSalary()[which is described below]*** . If ***TaxOnSalary*** instance is created without supplying value for any field, then value for ***salary*** and ***isPANsubmitted*** is by default initialized to ***0.0*** and ***false*** respectively.
(ii) Accessor methods(s) are provided for every instance field.
(iii) A method for computing the tax based on salary [***caculateTax() : double***] is supplied. The tax is calculated as per the rules shown below:
    a. if salary < 180000 and isPANsubmitted = true, then tax payable is zero

b. if salary < 180000 and isPANsubmitted = false, then tax payable is 5% of the salary
c. if 180000 < salary < 500000, then tax payable is 10% of the salary
d. if 500000 < salary < 1000000, then tax payable is 20% of the salary
e. if 1000000 < salary, then tax payable is 30% of the salary

(iv) A method named ***inputSalary()*** is supplied to read the value for the ***salary*** as an input from the user [**consider reading this value from keyboard**] and to assign the value to the corresponding instance variable ***salary***.

3.2 Write a Test class named TestTax.java which

a) Creates two instances of ***tax1*** and ***tax2*** of the class ***TaxOnSalary*** with different initializations [see point (i) in the description of Methods].
b) Takes salary as an input from the user [using keyboard] for both the instances ***tax1*** and ***tax2***.
c) Calculate and display tax for both the instance ***tax1*** and ***tax2***.