Lab Assignment 9-10

1 Background

This document contains assignments to be completed as part of the hands on for the subject Inroduction to Java programming and JEE framework.

Assignment 1: Local Variables

Objective: Understand the usage of local variables

Problem Description: To create different types of local variables and display them

Step 1: Type the below program in the textpad, save and compile the program

```
// Program to understand local variables and literals
class Variable {
      public static void main (String args[]){

      intintVal;
      floatfloatVal = 250.5f;
      doubledoubleVal = 2500.5;
      booleanboolVal = true;
      System.out.println("Integer\t:" + intVal + "\nFloat\t:" +
floatVal+"\nDouble\t:" + doubleVal + "\nBoolean\t:" + boolVal);
      }
}
```

Step 2: Observe that there is a compilation error. Debug the code so that it compiles and executes successfully.

Note: Accessing an uninitialized local variable will result in a compile-time error. If you cannot initialize your local variable where it is declared, make sure to assign a value before you attempt to use it.

Summary of this assignment:

In this assignment, you have learnt

- How to use local variables in Java

Assignment 2: Operators

Objective: Understand the various operators in Java

Problem Description: To implement and understand the usage of modulus operator, string concatenation, equality operator and logical operators in Java.

Step 1: Type the below program in the textpad, save, compile and execute

```
// Program to understand different types of operators

class Operator{
      public static void main(String[] args){
            intintVal=10;
            floatfloatVal=3.0f;
            boolean bool1 = true;
            boolean bool2 = false;
            boolean bool3 = true;


            // Arithmetic : Modulus operator
            System.out.println("Arithmetic");
            System.out.println(intVal + " % " + floatVal + "=" +
                              (intVal % floatVal));
            System.out.println();

            // String Concatenation : Observe the difference
            System.out.println("String Concatenation");
            System.out.println("Day " + 2 + " Session");
            System.out.println("\n" + 2 + 3 +"\n" +(2+3));
            System.out.println();

            // Relational: Equality operator
            System.out.println("Relational");
            System.out.println(intVal + " == " + floatVal + " = "
+(intVal==floatVal));
            floatVal = 10.0f;
            System.out.println(intVal + " == " + floatVal + " = "
+(intVal==floatVal));
            System.out.println();
```

```
            bool2 = false;

            if(bool1 || (bool1 && (bool2=false))){
                  System.out.println("Success");
            }
            else{
                  System.out.println("Failure");
            }
            System.out.println("bool2 value : " + bool2);
      }

}
```

Summary of this assignment:

In this assignment, you have learnt

- Usage of some operators available in Java

Assignment 3: Boolean Data Type

Objective: Understand the Boolean data type

Problem Description: To implement and understand the usage Boolean datatype in Java

Step 1: Type the below program in the textpad and save the file

Step 2: When executed, the output is :

Failure

bool2 value : true

Press any key to continue . . .

Step 3: Make changes to the code so that the output reads :

Success

bool2 value : true

Press any key to continue . . .

Note: The logical operators and the Boolean operators may be changed to achieve this result. Although the if control structure is used here, the focus is on the operators and Boolean datatype

```
class Operator{
	public static void main(String[] args){
		boolean bool1 = false;
		boolean bool2 = true;
		boolean bool3 = true;

		if(bool1 && (bool1 && (bool2=false))){
			System.out.println("Success");
		}
		else{
			System.out.println("Failure");
		}
		System.out.println("bool2 value : " + bool2);
	}
}
```

Summary of this assignment:

In this assignment, you have learnt

- Boolean data type and operators

Assignment 4: Control Statement

Objective: Understand control statements

Problem Description: To implement control flow statements break and continue

Step 1: Type the below programs in the textpad, save, compile and execute

```
Program 1:
//Program to understand the loop and break

class Control{
	public static void main(String args[]){
		booleanbool = true;

		for(int i= 0; i<5 ; i++){
			for(int j = 0; j<10; j++){
				System.out.print(j +"\t");
```

```
                        if(j > 5){
                              break;
                        }
                  }

            System.out.println("Outer Loop");
            }
            System.out.println("End");
      }

}
```

```
Program 2:
//Program to understand the loop and continue

classControlContinue{
      public static void main(String args[]){
            booleanbool = true;

            for(int i= 0; i<5 ; i++){
                  for(int j = 0; j<10; j++){
                        System.out.print(j +"\t");
                        if(j > 5){
                              System.out.println();
                              continue;
                        }
                  }

            System.out.println("Outer Loop");
            }
            System.out.println("End");
      }

}
```

Note:

- You can't use a number or anything that does not evaluate to a Boolean value as a condition in selection / iteration statement
- You can use break inside a looping statement without including if construct.

Summary of this assignment:

In this assignment, you have learnt

- How to use break and continue with the looping statement.

Assignment 5: Switch case statement

Objective: Understand the concept of conditional case statements using switch cases.

Problem Description: A java program needs to be written to display the range of marks based on the grade. The table for calculation of grade is as follows:

Grade	Range of marks
A	80-100
B	73-79
C	65-72
D	55-64
E	<55

Step 1: Create a class called SwitchCase with the main method

Step 2: Create a character variable for storing the grade with a default value of B

Step 3: Write the java code to implement the calculation of grades as given in the table. If any other grade is provided , the program must display a message “Grade does not exist”,for valid grades it must display the range of marks as given in the table.

Step 4: Change the values of the grade variable and test the working of the program for all the grades.

Summary of this assignment:

In this assignment, you have learnt

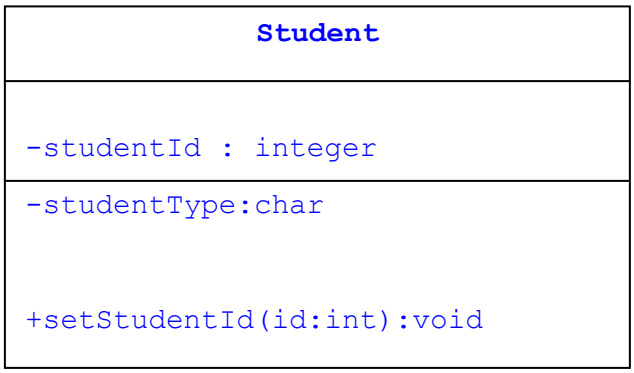
- Switch case statements

Assignment 6: Classes and Objects

Objective: Understand and implement the concept of encapsulation and abstraction.

Problem Description:

Implement the following class diagram using Java.



Note:

- Using a class , you can create any number of objects
- For every instance variable there should be a setter and getter method associated to it. Setter method is to set the value of the instance variable and getter method is to return the value of the instance variable
- studentType instance variable can have 'F' (fresher) or 'L' (lateral)

Step 1: Define a class "**Student**" and save it in Student.java

Step 2: Define all the member methods of Student Class

Step 3: Define a class "**CourseManagement**" with main method and save it in CourseManagement.java

a. In the main method, assign the student details in the corresponding temporary variables.
b. Create a reference variable with the name **student** and instantiate the same
c. Invoke the corresponding setter methods to set the instance variable with the given values stored in temporary variables.
d. Using getter methods, display the **student** details.

Step 4: Compile the program, fix the errors if any

Step 5: Execute the program and verify the output

Summary of this assignment:

In this assignment, you have learnt

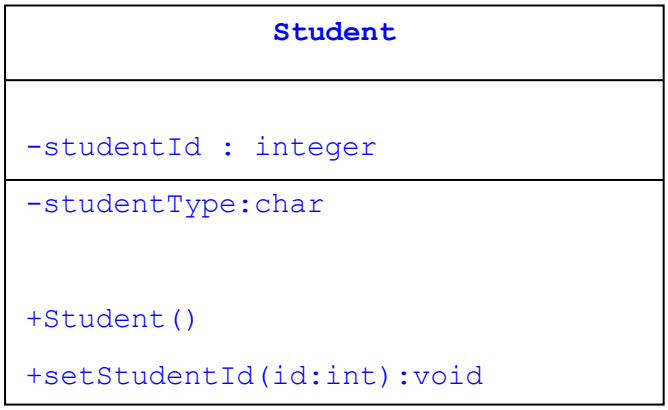
- How to implement getter and setter methods
- How to create reference variables
- How to create an object

Assignment 7: Default Constructor

Objective: Understand the concept of Default Constructor.

Problem Description:

Enhance the Assignment 4 and do the necessary modification as mentioned below:



Step 1: Define constructor in Student class

Student (): initialize the instance variables with 10 and 'F' (indicated fresher) respectively

Step 2: Define a main method

a. In the main method, create a reference variable with the name **student**.
b. Create an object of type **Student** and assign it to reference variable named **student**.
c. Using getter methods, display the **student** details.

Step 3: Compile the program, fix the errors if any

Step 4: Execute the program and verify the output

Estimated time: 15 mins

Summary of this assignment:

In this assignment, you have learnt

- How to implement default constructor
- How to use the constructor for initializing the objects

Assignment 8: Default and Parameterized constructors

Objective: Understand the concept and working of default and parameterized constructor.

Problem Description: Create a class called UserType and create a default and a parameterized constructor. Invoke these constructors through two different objects of the UserType class.

```
public class UserType {

                String name;

      UserType(String parameterVal)
             {
      name = parameterVal;
             }

      UserType()
             {
      name = "Student";
             }

      public static void main(String args[]) {
            UserType usertype1 = new UserType("Faculty");
            UserType usertype2 = new UserType();

            System.out.println(usertype1.name);
            System.out.println(usertype2.name);
              }

}
```

Summary of this assignment:

In this assignment, you have learnt about

- Default and
- Parameterized constructors

Optional Assignment 9: ElectricityBill with getter and setter methods

Objective: To write a program with getter and setter methods.

Problem Description: Create a class called ElectricityBillwhich has

1. Four data members customerId (int), customerName(String), previousReading(float), currentReading (float).

2. Introduce setter and getter methods for assigning and retrieving values for the instance variables.

3. Write a public method called printDetails with the following prototype
   public void printDetails() which prints all the details of the ElectricityBill such as customerId, customerName, NoOfUnits.

   (Hint :NoOfUnits = currentReading – previousReading)

4. Write a main method which creates an object of ElectricityBillinvokes the setter and getter methods And also prints the electricity bill details by calling printDetails method

Assignment 10: this keyoword

Objective: To write a Java class to understand the concept of this keyword.

Problem Description: Customer class is created such that the name of method arguments and the name of instance variables are one and the same.

Step 1: Create a folder Assignment1 under your work directory

Step 2: Create a Customer class as given below

FileName: Customer.java

```
/*
 * This java file contains a class having the attributes of a
 * Customer and has a parameterized constructor to initialize the
 * instance variables
 */

/**
 * This class has four attributes customerId, customerName,
 * customerTelNo, customerEmail and uses the parameterized
constructor
 * to initialize the instance variables
 * Date: 16-Mar-2007
*
* @version 1.0
 */

class Customer
{
      privateintcustomerId;
      private String customerName;

      /**
       * Parameterized Constructor: Creates a customer Object and
* initializes the instances variables with the supplied values.
       * Since the method arguments name and instance variable names
       * are similar we have to use this keyword which has a
       * reference to the current object
       */
      Customer(intcustomerId, String customerName)
      {
            this.customerId= customerId;
            this.customerName= customerName;
      }

      /**
      *This method displays the customer details
      *@param void
      *@return void
      */
      public void displayCustomer()
      {
```

```
      System.out.println("Customer Id  :" + customerId);
      System.out.println("Customer Name :" + customerName);
      }

      /**
       * Creates an object of the Customer class and
       * and invokes the method displayCustomer using the object to
       * print the customer details
       * @paramargs Command line arguments
       */

      public static void main(String argv[])
      {
            Customer c1=new Customer(1001, "John");
      c1.displayCustomer();
      }
}
```

Step4: Compile and execute the above program. The output should come up.

Customer Id: 1001

Customer Name: John

Step 5: Modify the program by eliminating the "this" keyword in all the statements inside the parameterized constructor. Save the changes and analyze the results after execution.

Summary of this exercise:

You have just learnt

- How to write parameterized constructor in java using this keyword.

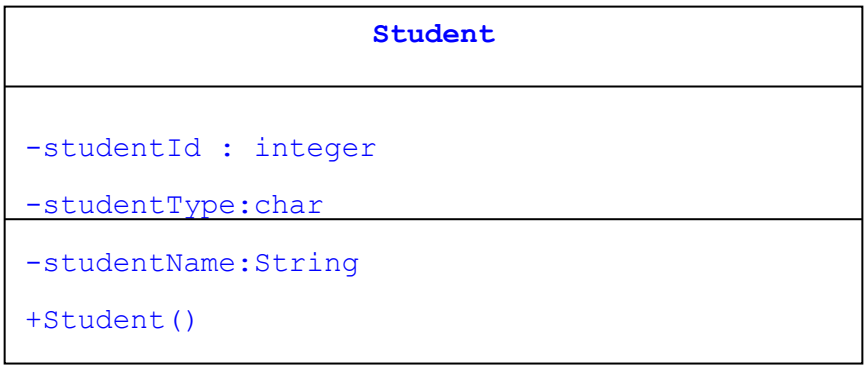
Note: this keyword acts as a reference to the current object with which the parameterized constructor is invoked.

this keyword must be used within a parameterized constructor when the instance variables' name and arguments name are same.

Assignment 11: Reference variable as an argument to a method

Objective: Understand the usage of "this" keyword.

Problem Description: Implement the class diagram Using Java



Step 1: Define a class Student ,

- Define two constructors
  - Default constructor: Used to increment the value of studentId for each objects of Student class. studentId should start from 550.
  - Parameterized constructor: Constructor with three arguments StudentType(H/D),firstName and lastName
    studentName=firstName+lastName
  - displayDetails (Student obj):Should accept the object as an argument and display the details like studentId, studentType, studentName of that object

Step 2: Define a main method ,

- In the main method
  - Create an object of Student('D',"Bony","Thomas") and assign it to reference variable studentOne;
  - Call the displayDetails() method and display the details of studentOne
  - Create an object of Student('H',"Dinil","Bose") and assign it to reference variable studentTwo;
  - Call the displayDetails() method and display the details of studentTwo

Step 3: Compile the program, fix the errors if any

Step 4: Execute the program and verify the output

Summary of this assignment:

In this assignment, you have learnt

- How to use this, passing object to an argument, Overloaded constructors

Assignment 12: User Input(Command Line Arguments)

Objective: To work with command line arguments.

Step 1: Create a folder Assignment3 under your work directory

Step 2: Open a text editor, notepad or Dos Editor and type the following:

```
/* This file is a demo Java program which depicts the concept of
 * command line arguments
 */

/**
 * This class contains the main method which accepts the
 * two integer values as command line arguments and displays the
sum.
* Date: 20-Apr-2007
 *
* @version 1.0
 */

public class CommandLineArgs  {

/**
      * Starting point of the application.
      * Expects exactly two command line arguments else
      * prints an error message and terminates the program
      * otherwise it will parse the arguments to integers and
      * displays the sum by adding them
      * @paramargs Command line arguments
      */
     public static void main(String args[])
     {

           if (args.length< 2 || args.length> 2)
           {
                System.out.println("Invalid no of arguments"+
                                   "Supply exactly two arguments");
                System.exit(0);
           }
           System.out.println(args[0]+" "+args[1]);
int x=Integer.parseInt(args[0]);
           int y=Integer.parseInt(args[1]);
           System.out.println(x+y);
     }
}
```

Step 3: Save the file as 'CommandLineArgs.java'

Step 4:Compiling the program. Close the editor. Now compile your program using the command line:

```
javac CommandLineArgs.java
```

Step 5: Run your program using the command line:

```
javaCommandLineArgs
```

Check the output of the program.

Now run the program using the command line:

```
javaCommandLineArgs 5 6
```

Now the output will be 5 6 and 11

Hint: Refer to Javadoc for the wrapper classes

Summary of this exercise:

You have just learnt

- How to pass command line arguments into a java program
- The arguments are passed as array of Strings.
- How to convert the String value to its respective integer value in order to perform numerical operation on it

Assignment 13: Java Doc Exploratory assignment on Strings

Objective: To explore and understand the Java Doc. Create a program with the help of java doc. (hint use String)

Problem Description: Create classes called which does the following:

1. Takes User name as input
2. Display length of the string
3. Displays output as Hi <username>
4. Convert User name to lower case
5. Convert User name to upper case

6. Check whether the name is starting with ‘a’

Summary of this assignment:

In this assignment, you have learnt

- To use Java Doc to seek help on existing classes

Assignment 14: Static variable, block and method

Objective: To write a Java class to understand the concept of static keyword.

Problem Description: To the Customer class created in Assignment 1 of Day 2, modify the Customer class so as to help us to keep track of the number of Customer objects created during runtime.

Step 1: Create a folder Assignment5 under your work directory

Step 2: Copy the Customer.java file created in Assignment 1 of Day 2 into this folder.

Step 3: Do the necessary modifications as conveyed below.

Filename: Customer.java

```
/**
 * This java file contains a class having the attributes of a
 * Customer and has a parameterized constructor to initialize the
 * instance variables.
 * Demonstrates the usage of static block, static variables and
static
 * methods
 */

/**
 * This class has four attributes customerId ,customerName,
 * customerTelNo, customerEmail and uses the parameterized
constructor
 * to initialize the instance variables
 * Also declare a static variable named noOfCustomers which help us
 * to keep track of the number of customer objects created during
* runtime.
 * Date: 20-Apr-2007
 *
* @version 1.0
```

```
 */

class Customer
{

      /* Static block - A block of statements written to
       * initialize static variables
       * The statements inside the static block will get
       * executed only once when the class is loaded
       * by the JVM
       */

      static
      {
      /* Assume that we already have 50 customers. Hence the
       * static variable noOfCustomers has to be initialized
       * to 50
       */
noOfCustomers = 50;
      }

      /* declaration of instance variables
       * these variables are available for every objects created
       */
privateintcustomerId;
private String customerName;
privateintcustomerTelNo;
private String customerEmail;


      /* declaration of static variables (class variables )
       * this is common for all the objects created
       */

staticintnoOfCustomers;

      /**
       * Parameterized Constructor: Creates a customer Object and
* initializes the instances variables with the supplied values.
       * Since the method arguments name and instance variable names
       * are similar we have to use this keyword which has a
       * reference to the current object
       */

      Customer(intcustomerId, String customerName, intcustomerTelNo,
                                              String customerEmail)
      {
            this.customerId= customerId;
            this.customerName= customerName;
```

```
this.customerTelNo= customerTelNo;
this.customerEmail= customerEmail;

           /* increment the static variable by one since
            * we have created one customer object */

noOfCustomers++;
      }

      /**
      *This method displays the customer details
      *@param void
      *@return void
      */

      public void displayCustomer()
      {
      System.out.println("Customer Id  :" + customerId);
      System.out.println("Customer Name :" + customerName);
      System.out.println("Customer TelNo : " + customerTelNo);
      System.out.println("Customer Email Id : " + customerEmail);
      }

      /**
      *This method displays the count of
     * the number of customer objects created during runtime.
     * This is static method (class method) which can access
     * only static variables
      *@param void
      *@return void
      */

      public static void customerCount()
      {
      System.out.println("No of Customers : "+ noOfCustomers);
      }

      /**
       * Creates an object of the Customer class and
       * and invokes the method displayCustomer using the object to
      * print the customer details
       * @paramargs  Command line arguments
       */

      public static void main(String argv[])
      {
//creating customer object one
            Customer c1=new Customer(1001, "John", 465364,
           "John@yahoo.com");
```

```
      //creating customer object two
            Customer c2=new Customer(1002, "Jack", 764755,
            "Jack@yahoo.com");

      //display the count of customer objects
      Customer.customerCount();

      // c2.displayCustomer();
      }
}
```

Step 4: Compile and execute the program. Analyze the results.

Step 5: Omit the class name alone and just say **customerCount()** alone within the code while invoking the static method. Compile the program and analyze the results.

Step 6: Try invoking the static variable **noOfCustomers**within a non-static method **displayCustomer()** and remove the comments in the last statement of the main method.

Compile the program and analyze the results.

Summary of this exercise:

You have just learnt

- When to go for static variables, and how to declare static variables
- When to go for static methods, and how to declare and implement static methods
- When to go for static block, and how to declare the same.

Note:

Static block: Used to initialize static variables. Statements inside the static block are executed only once when the class loaded by the JVM.

Static variable (Class variables) – A single copy of the variable is retained and is shared by all objects. Static numeric variables are Initialized to zero.

Static methods (Class methods) can access only static variables.