East West University
Department of Computer Science and Engineering
A/2, Jahurul Islam Avenue, Jahurul Islam City, Aftabnagar, Dhaka

Lab Manual: 01

Lab Topic: Introducing Basics of Elementary Programming in Java

Course Code: CSE110 (Object Oriented Programming)

Course Instructor: Tanni Mittra, Senior Lecturer, CSE

Lab Objective

- **Understand** basic program structure in Java
- **Solve** a few simple problems in Java.
- **Solve** problems using selection statement

Lab Activities

A. Reading inputs from user

- Everything in Java comes in form of a class.
- To read inputs from a user, we need to use *Scanner* class.
- The following program reads name, age and department name of a student and print them accordingly.

```
import java.util.Scanner;
class SampleReadInput{
 public static void main (String[] args){
    Scanner input = new Scanner (System.in);
    System.out.println("Enter your name: ");
    String name = input.next();
    System.out.println("Enter your age: ");
    int age = input.nextInt();
    System.out.println("Enter your CGPA: ");
    double cgpa = input.nextDouble();
    System.out.println("Enter your department: ");
    String department = input.nextLine();
    System.out.printf("Your Name: %s\n", name);
    System.out.printf("Your Age: %d\n", age);
    System.out.printf("Your CGPA: %f\n", cgpa);
    System.out.printf("Your Deparmtent: %s\n", department);
  } // main method ends
} // Main class ends
```

- **Does the program execute as we have wanted? What is the problem? How can you solve it?**

Lab Problems

01: Write a program that reads an integer from the console and determines whether the given number is divisible by either 2 or 3 (but not both). Then the program should print TRUE, otherwise, the program should print FALSE.

02: Write a program that prompts the user to enter the minutes (e.g., 1 billion), and displays the number of years and days for the minutes. For simplicity, assume a year has **365** days. Here is a sample run:

```
Enter the number of minutes: 1000000000 ↵Enter
1000000000 minutes is approximately 1902 years and 214 days
```

03: The two roots of a quadratic equation $ax^2 + bx + c = 0$ can be obtained using the following formula:

$$r_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a} \quad \text{and} \quad r_2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a}$$

$b^2 - 4ac$ is called the discriminant of the quadratic equation. If it is positive, the equation has two real roots. If it is zero, the equation has one root. If it is negative, the equation has no real roots. Write a Java program that prompts the user to enter values for *a*, *b*, and *c* and displays the result based on the discriminant. If the discriminant is positive, display two roots. If the discriminant is 0, display one root. Otherwise, display "The equation has no real roots".

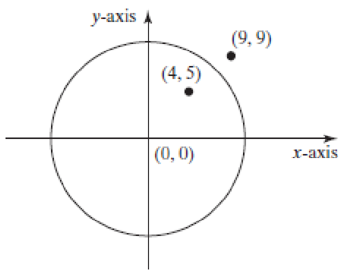
Note that you can use **Math.pow(x, 0.5)** to compute 2*x*.

```
Enter a, b, c: 1.0 3 1 ↵Enter
The equation has two roots -0.381966 and -2.61803
```

```
Enter a, b, c: 1.0 3 1 ↵Enter
The equation has two roots -0.381966 and -2.61803
```

04: Write a Java program that prompts the user to enter the exchange rate from currency in U.S. dollars (USD) to Bangladeshi Taka (BDT). Prompt the user to enter **0** to convert from USD to BDT and **1** to convert from BDT to USD. Then, prompt the user to enter the amount in USD or in BDT to convert it to BDT or USD, respectively.

05. Write a Java program that prompts the user to enter the center (**p**,**q**) and the radius (**r**) of a circle. Then, prompts user to enter a point (**x**, **y**) and checks whether the point is within the circle centered at (**p, q**) with radius **r**. For example, (**4**, **5**) is inside the circle centered at (0,0) with radius 10 and (**9**, **9**) is outside the circle, as shown in the following figure.



East West University
Department of Computer Science and Engineering
A/2, Jahurul Islam Avenue, Jahurul Islam City, Aftabnagar, Dhaka

Lab Manual: 02

Lab Topic: Loops and Basic Array Manipulations

Course Code: CSE110 (Object Oriented Programming)

Course Instructor: Tanni Mittra, Senior Lecturer, CSE

Lab Objective

1. **Apply** concepts of loops and arrays
2. **Write** and **execute** programs using these concepts in Java.

Lab Activities

A. Arrays in Java

- Arrays are declared in the following manner in Java.

```
int[] numbers = new int[10];
```

- The above-mentioned line created an integer array numbers of size 10.
- The array elements can be accessed in the same way as in C.
- Examine the following program.

```
  import java.lang.*;
  import java.util.*;
  class SampleArray{
   public static void main(String[] args){
     // initializing an integer array numbers with size 10
     int[] numbers = new int [10];
     Scanner input = new Scanner (System.in);
     // assigning values randomly
     for(int i=0; i<numbers.length; i++){
          numbers[i] = input.nextInt();

          // numbers[i] = (int)(Math.random()*100);// Read random numbers
     }
     // displaying the values
     for(int i=0; i<numbers.length; i++){
          System.out.print(numbers[i] + " ");
     }
   }
}
```

- **numbers.length** returns the size of the array.
- For Java String, you can use **chatAt(i)** method to access the character at ithposition of the string. Suppose,

```
String str = "Hello World";
System.out.print(str.charAt(0)); // returns the character 'H'
```

B. Enhanced For loop

- Java also includes another version of for loop
- Enhanced for loop provides a simpler way to iterate through the elements of a collection or array
- It is read only loop where you can’t update the values as opposite to other loops

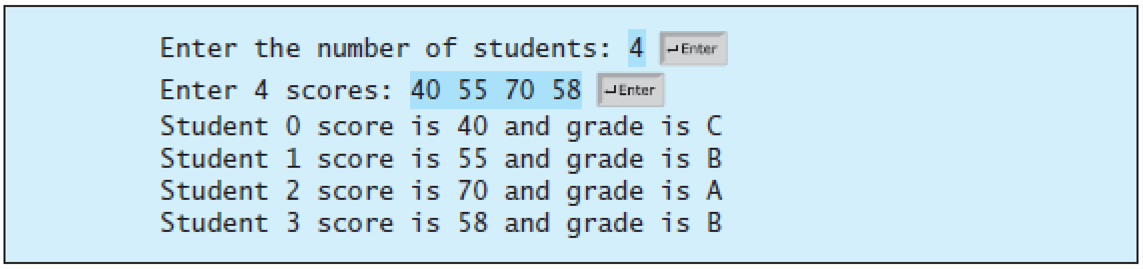
```
public class enhancedforloop
{
    public static void main(String args[])
    {
        int array[] = {10, 20, 30)

        //enhanced for loop
        for (int x:array)
        {
            System.out.println(x);
        }

        /* for loop for same function
        for (int i = 0; i < array.length; i++)
        {
            System.out.println(array[i]);
        }
        */
    }
}
```

Lab Problems

01: Write a Java program that reads student scores, gets the best score, and then assigns grades based on the following scheme:

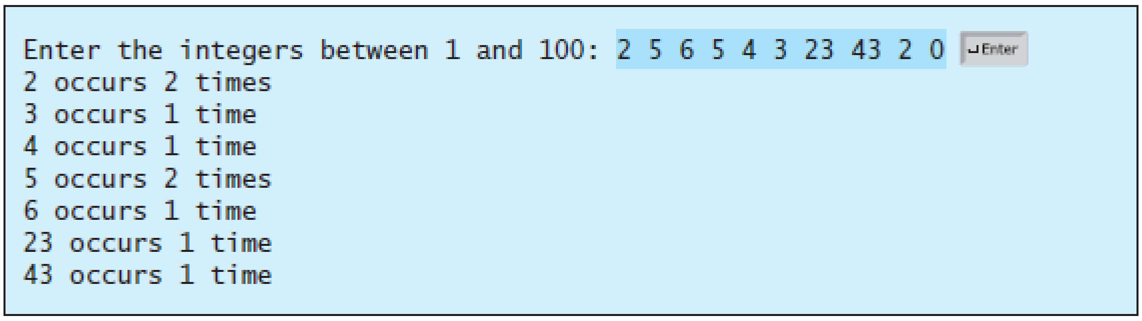
Grade is A if score is ≥ best - 10
Grade is B if score is ≥ best - 20;
Grade is C if score is ≥ best - 30;
Grade is D if score is ≥ best - 40;
Grade is F otherwise.

The program prompts the user to enter the total number of students, then prompts the user to enter all of the scores and concludes by displaying the grades. Here is a sample run:



```
Enter the number of students: 4 ↵Enter
Enter 4 scores: 40 55 70 58 ↵Enter
Student 0 score is 40 and grade is C
Student 1 score is 55 and grade is B
Student 2 score is 70 and grade is A
Student 3 score is 58 and grade is B
```

02: Write a Java program that reads the integers between 1 and 100 and counts the occurrences of each. Assume the input ends with **0**. Here is a sample run of the program:



```
Enter the integers between 1 and 100: 2 5 6 5 4 3 23 43 2 0
2 occurs 2 times
3 occurs 1 time
4 occurs 1 time
5 occurs 2 times
6 occurs 1 time
23 occurs 1 time
43 occurs 1 time
```

03: A *Palindrome* is a number that reads the same from either way (forward or backward). As an example, 1221 is a palindrome, but 123 is not. Write a Java program that prompts the user to enter a number and displays whether the number is a palindrome or not.

04: Write a Java program that reads in ten numbers and displays the number of distinct numbers and the distinct numbers separated by exactly one space (i.e., if a number appears multiple times, it is displayed only once). (*Hint*: Read a number and store it to an array if it is new. If the number is already in the array, ignore it.) After the input, the array contains the distinct numbers. Write a method Isdistinct() to check whether a number exists multiple times or not.

05: Write a Java program that randomly generates an integer array, *numbers*, of size 100. Then, find the value and index (position) of the highest and the smallest element. Use separate method for determining highest and smallest element.

06: Write a program that prompts the user to enter the number of students, the students' names, and their scores, and prints student names in decreasing order of their scores.

East West University
Department of Computer Science and Engineering
A/2, Jahurul Islam Avenue, Jahurul Islam City, Aftabnagar, Dhaka

---

Lab Manual: 03

Lab Topic: String and Two-Dimensional Array Manipulations

Course Code: CSE110 (Object Oriented Programming)

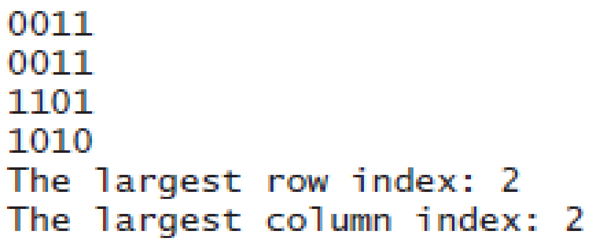
Course Instructor: Tanni Mittra, Senior Lecturer, CSE

Lab Objective

1. **Apply** concepts of two dimensional arrays and String
2. **Write** and **execute** programs using these concepts in Java.

Lab Problems

01: Write a program that randomly fills in 0s and 1s into a 4-by-4 matrix, prints the matrix, and finds the first row and column with the most 1s. Here is a sample run of the program:



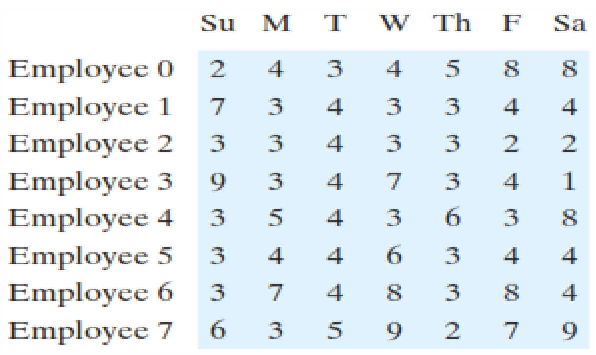
```
0011
0011
1101
1010
The largest row index: 2
The largest column index: 2
```

02: Write a method that checks whether a string is a valid password. Suppose the password rules are as follows:

- A password must have at least eight characters.
- A password consists of only letters and digits.
- A password must contain at least two digits.

Write a program that prompts the user to enter a password and displays a Valid Password if the rules are followed or an Invalid Password otherwise.

03: Suppose the weekly hours for all employees are stored in a two-dimensional array. Each row records an employee's seven-day work hours with seven columns. For example, the following array stores the work hours for eight employees. Write a program that displays employees and their total hours in decreasing order.



	Su	M	T	W	Th	F	Sa
Employee 0	2	4	3	4	5	8	8
Employee 1	7	3	4	3	3	4	4
Employee 2	3	3	4	3	3	2	2
Employee 3	9	3	4	7	3	4	1
Employee 4	3	5	4	3	6	3	8
Employee 5	3	4	4	6	3	4	4
Employee 6	3	7	4	8	3	8	4
Employee 7	6	3	5	9	2	7	9

04: Write a Java program to Sort N number of strings entered by keyboard in Lexicographical Order (Dictionary

Order).

```
Enter 5 words: R programming
JavaScript
Java
C programming
C++ programming

In the lexicographical order:
C programming
C++ programming
Java
JavaScript
R programming
```

05: Write a program to sort a two-dimensional array according to the values in any given column

```
Input: If our 2D array is given as (Order 4X4)

    39 27 11 42
    10 93 91 90
    54 78 56 89
    24 64 20 65
    Sorting it by values in column 3
Output: 39  27  11  42
        24  64   20 65
        54  78   56  89
        10   93   91 90
```

East West University
Department of Computer Science and Engineering
A/2, Jahurul Islam Avenue, Jahurul Islam City, Aftabnagar, Dhaka

Lab Manual: 04

Lab Topic: Introducing Classes and Objects

Course Code: CSE110 (Object Oriented Programming)

Course Instructor: Tanni Mittra, Senior Lecturer, CSE

Lab Objective

1. **Familiarize** students with the implementation of classes
2. **Instantiation** of objects accordingly.

Lab Activities

A. Writing Class Definition

- Define a class 'Icecream' that has the following instance variables:
    a. icecreamType (String)
    b. icecreamCompany (String)
    c. icecreamPrice (double)
- At the beginning, the access specifier is default (no public or private while declaring them).
- Define appropriate constructors for 'Icecream' class that set the attributes values. One constructor should take attributes value as parameter.
- Define the following instance methods:
    a. toString(): returns a string containing an icecream information.
    .

B. Demonstrating Class Functionalities

- Define the Main (Driver) class that has the main() method.
- Create two icecream objects using both constructors.
- Print their values.

C. Introducing private access specifier and Setters (Mutators) and Getters (Accessors)

- While declaring instance variables, using private access modifier indicates that the instance variable cannot be accessed from outside the class definition. It is helping to achieve data hiding that states that the data of an object must not be accidentally modified or updated.
- Therefore, to access the private instance variables from outside the class, we need to include a few more instance methods known as setters and getters.
- **Setters are used to set the value.**

```
void setIcecreamType(String icecreamType){
   icecreamType = icecreamType;
}
```

- **Getters are used to return the value.**

```
String getIcecreamType(){
   Return icecreamType;
}
```

- **Modify your Icecream class definition accordingly by including setters and getters.**
- Remove constructors for 'Icecream' class in previous example and define new constructor that set price of the Icecream class to zero.

D. Passing Objects to Methods

- In Java we can pass objects to methods
- When we pass a primitive type to a method, it is passed by value.
- But when we pass an object to a method, the situation changes dramatically, because objects are passed by what is effectively call-by-reference.
- **Add equals (Icecream I) method to your Icecream class and return true if price of caller object and callee object are same; false, otherwise.**

- **Add compareTo(Icecream I): returns 1 if price of caller object is higher, 0 if both prices are the same, -1 otherwise**

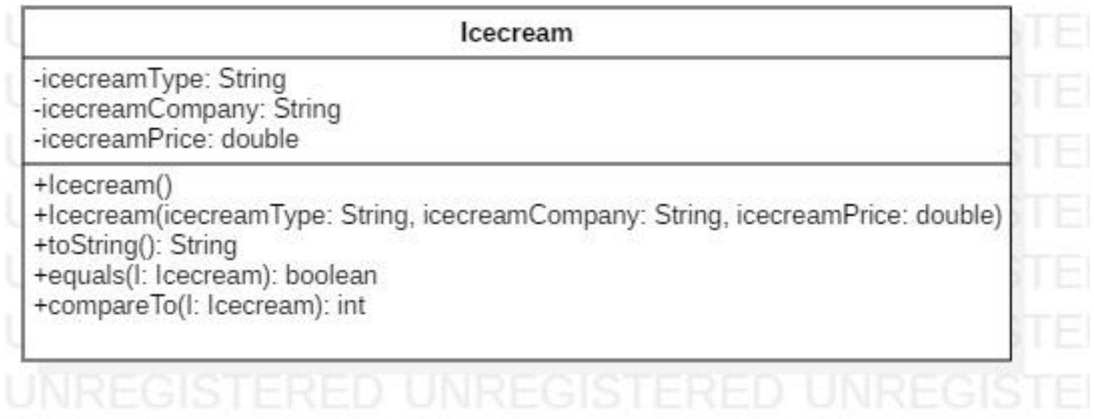
E. Array of Objects

- An array of objects is actually an array of reference variables.
- For the Icecream class we can create array of objects :

Icecream[] IcecreamArray = new Icecream [10];

F. Introducing Class Diagram

- Specifications of a class can be also expressed using a class diagram.
- Class diagram is a standard way to represent a class.
- The above-mentioned Icecream class can be represented using the following diagram.


Icecream
-icecreamType: String
-icecreamCompany: String
-icecreamPrice: double
+Icecream()
+Icecream(icecreamType: String, icecreamCompany: String, icecreamPrice: double)
+toString(): String
+equals(I: Icecream): boolean
+compareTo(I: Icecream): int
UNREGISTERED UNREGISTERED UNREGISTE

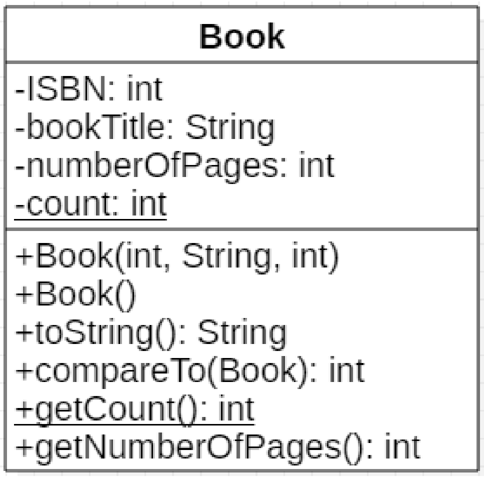
Lab Problems

01: Based on the 'Icecream' class as defined before, now create an array of Icecream type objects and get the input from the user to initialize those objects. The size of array must be at least 5.

Now, write a static method searchByCompany in the Main class of the previous implementation, which takes a String parameter representing company name and then prints all icecream's information manufactured by that company.

02: Write a program in Java that follows the specifications as given below.

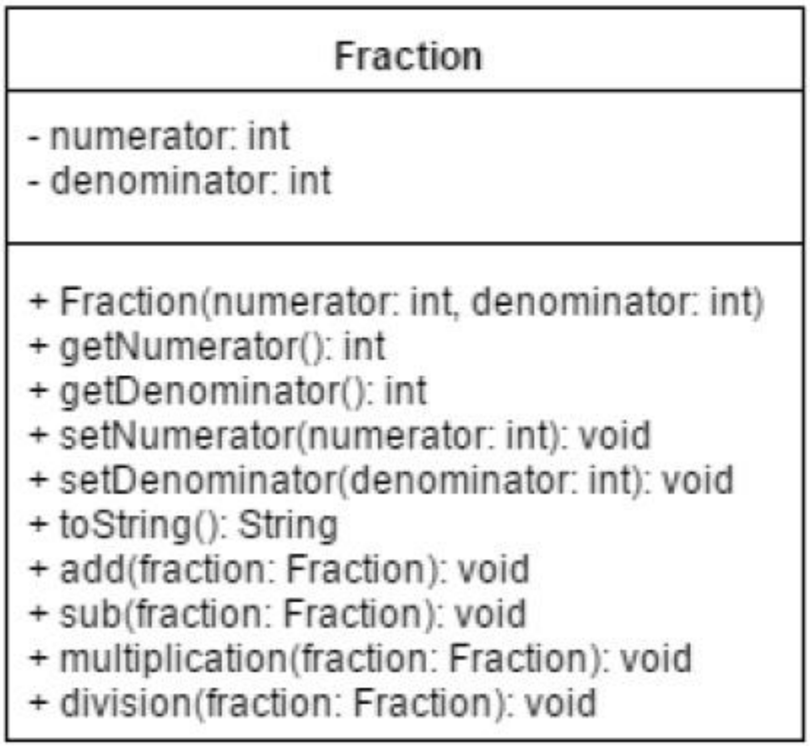
- Define a class **Book** as shown in the following figure.



- Create your *Main* class and the *main()* method. Define an array of Book type objects of size 5.
- Instantiate these Book objects by taking the inputs from the user.

- Print all Book objects' data using a static method *displayAll()*.
- Invoke *compareTo()* method to compare any two Book objects based on their pages. If caller object's number of pages is greater than callee object's number of pages, compareTo() returns 1; if both pages are same, the method returns 0 and -1 otherwise. Make sure to print the returned value.
- Define a static method *isHeavier()* within the *Main* class that takes a Book object as input parameter and returns true if the Book's number of pages is greater than 500; false, otherwise. Add appropriate code into your main() method to demonstrate its functionalities.

03: Implement the following class 'Fraction' and test its methods.



```
Fraction

- numerator: int
- denominator: int

+ Fraction(numerator: int, denominator: int)
+ getNumerator(): int
+ getDenominator(): int
+ setNumerator(numerator: int): void
+ setDenominator(denominator: int): void
+ toString(): String
+ add(fraction: Fraction): void
+ sub(fraction: Fraction): void
+ multiplication(fraction: Fraction): void
+ division(fraction: Fraction): void
```

void add(Fraction fraction)
Adds two Fraction objects and **stores the result** into **calling object**. This is how addition is performed for fractions:
1 / 4 + 3 / 5 = 1 * 5 + 3 * 4 / 4 * 5 = 17 / 20

String toString()
Returns the value of the fraction in 1 / 2 format where 1 is numerator and 2 is denominator.

Now write a test program, take two Fraction objects. Print both of them. Test add, sub, multiplication and division methods. Print calling object after each method call.

East West University
Department of Computer Science and Engineering
A/2, Jahurul Islam Avenue, Jahurul Islam City, Aftabnagar, Dhaka

Lab Manual: 05

Lab Topic: Class Relationships

Course Code: CSE110 (Object Oriented Programming)

Course Instructor: Tanni Mittra, Senior Lecturer, CSE

Lab Objective

1. **Familiarize** students with the implementation of classes
2. **Write** various instance methods performing different actions on the objects of a class
3. **Write** the definition of multiple classes and
4. Can **identify** and **hold** their relationships among each other.

Problem_1: Your job is to define the above-mentioned classes as per the specification mentioned below and then write a Main/Driver class that demonstrates the functionalities of these classes.

Student	**Course**
- studentId: int -studentName: String -studentCGPA: double -clist[]: Course -numberofcourse:int	- courseId: String - courseTitle: String - credit: double - studentList[]: Student - numberOfStudents: int
+ Student() + Student(studentId, studentName, studentCGPA) + display(): void + addCourse(Course): void + printCourseList(): void	+ Course() + Course(courseId, courseTitle, credit) + display(): void + addStudent(Student): void + printStudentList(): void

Now create a main class that will create objects of Student and Course class. Display list of courses taken by student and also show list of students of a course.

East West University
Department of Computer Science and Engineering

CSE 110: LAB 06
Course Instructor: Tanni Mittra

Inheritance and Polymorphism

Lab Objective

Familiarize students with the implementation of inheritance and polymorphism in Java.

Lab Outcomes

After completing this lab successfully, students will be able to:
1. **Write** the definition of the super class and extend it to create multiple subclasses.
2. **Write** codes to implement polymorphism.

Lab Activities

A. Defining the Superclass

(*The **Account** class*)
Design a class named **Account** that contains:

- A private **int** data field named **id** for the account (default **0**).
- A private **double** data field named **balance** for the account (default **0.0**).
- A private **double** data field named **annualInterestRate** that stores the current interest rate (default **0.0**).
- A private **Calendar** data field named **dateCreated** that stores the date when the account was created. Use Calendar type object.
- A no-arg constructor that creates a default account.
- A constructor that creates an account with the specified id, initial balance and annual interest rate. Within the constructor, assign the value of dateCreated using Calendar.getInstance().
- The accessor and mutator methods for **id**, **balance**, and **annualInterestRate**.
- The accessor method for **dateCreated**.
- A method named **getMonthlyInterestRate()** that returns the monthly interest rate. **monthlyInterestRate** is **annualInterestRate / 12**. Note that **annualInterestRate** is a percentage, e.g., like 4.5%. You need to divide it by 100
- The method **getMonthlyInterestAmount()** is to return monthly interest amount, not the interest rate. Monthly interest amount is **balance * monthlyInterestRate**.
- A method named **withdraw** that withdraws a specified amount from the account.
- A method named **deposit** that deposits a specified amount to the account.

Write a test program that creates an **Account** object with an account ID of 1122, a balance of $20,000, and an annual interest rate of 4.5%. Use the **withdraw** method to withdraw $2,500, use the **deposit** method to deposit $3,000, and print the balance, the monthly interest, and the date when this account was created.

B. Creating the Subclasses

Create two subclasses for checking and saving accounts named as **CheckingAccount** and **SavingsAccount**.

- A checking account has an overdraft limit which is double type variable.
- A savings account has issued with a credit card automatically. It holds the 16-digit card number an expiry date (Calendar type object). SavingsAccount class must have a method getCreditBalance that returns a credit balance which is three times of the current balance in the account.

C. Defining an Array/ArrayList of Account type objects in Main method

To understand polymorphism, you need to define an array/array list of Account type objects based on user-provided option. Your main() method must display the following first:

```
Press (1) for creating a Checking Account
Press (2) for creating a Savings Account
```

You must create at least 4 account type objects in this manner and perform one deposit and one withdraw operation for each account.

Afterwards, print the followings for each account using the concept of Polymorphism. You must not use any toString() method.

For a Checking Account:	For a Savings Account:
This is a Checking Account Account ID: Date Created: Current Balance: Annual Interest Rate: Monthly Interest Amount: Overdraft Limit:	This is a Savings Account Account ID: Date Created: Current Balance: Annual Interest Rate: Monthly Interest Amount: Credit Card Number: Card Expiry Date: Credit Balance:

East West University
Department of Computer Science and Engineering
A/2, Jahurul Islam Avenue, Jahurul Islam City, Aftabnagar, Dhaka

Lab Manual: 07

Lab Topic: Exception Handling

Course Code: CSE110 (Object Oriented Programming)

Course Instructor: Tanni Mittra, Senior Lecturer, CSE

Lab Objective

1. **Learn a** mechanism to handle Exception in Java program

Lab Activities:

A. Built-in Exceptions Handle

```
class ArithmeticException_Demo
{
    public static void main(String args[])
    {
        try {
            int a = 30, b = 0;
            int c = a/b;
            System.out.println ("Result = " + c);
        }
        catch(ArithmeticException e) {
            System.out.println (e);
            System.out.println ("Can't divide a number by 0");
        }
    }
}

Class Testthrows1 {
  Void m () throws IOException
  {
    throw new IOException("device error");//checked exception
  }
}
```

Lab problem 1:

- Write a program that creates a *Calculator* class. The class contains two variables of integer type. Design a constructor that accepts two values as parameter and set those values.
- Design four methods named *Add (), Subtract (), multiply (), Division ( )* for performing addition, subtraction, multiplication and division of two numbers.
- For addition and subtraction, two numbers should be positive. If any negative number is entered then throw an exception in respective methods. So design an exception handler (*ArithmeticException*) in *Add ()* and *Subtract ()* methods respectively to check whether any number is negative or not.
- For division and multiplication two numbers should not be zero. If zero is entered for any number then throw an exception in respective methods. So design an exception handler (*ArithmeticException*) in *multiply ()* and *Division ()* methods respectively to check whether any number is zero or not.
- Write a main class and declare four objects of *Calculator* class. Perform addition (obj1), subtraction (obj2), multiply (obj3) and division (obj4) operations for these objects. If any non integer values are provided as input; then you should throw an exception (*NumberFormatException*) and display a message that informs the user of the wrong input before exiting.

Lab problem 2:

- Create an exception class named *MyException* that extend a base class named *Exception*
- Design a constructor in your class that accepts a string value set it to the super class constructor to display the exception message.
- Create a main class named *product*. Write a method inside the class called *productCheck(int weight)* that accepts weight of the product. Inside the method, if the weight is less than 100 then throw an exception “Product is invalid” otherwise print the weight of the product.
- Inside the main method declare single object of the product class and call the *productCheck()* method to display the weight of the product.

Lab problem 3:

- Write a program that meets the following requirements: Creates an array with 100 randomly chosen integers. Prompts the user to enter the index of the array, then displays the corresponding element value. If the specified index is out of bounds, display the message Outof Bounds.

Lab problem 4:

- Design a class named Triangle that extends GeometricObject. The class contains: Three double data fields named side1, side2, and side3 with default values 1.0 to denote three sides of the triangle. A no-arg constructor that creates a default triangle. A constructor that creates a triangle with the specified side1, side2, and side3. The accessor methods for all three data fields. A method named getArea() that returns the area of this triangle. A method named getPerimeter() that returns the perimeter of this triangle. A method named toString() that returns a string description for the triangle. The toString() method is implemented as follows: return "Triangle: side1 = " + side1 + " side2 = " + side2 +" side3 = " + side3;
- Write a test program that prompts the user to enter three sides of the triangle, a color, and a Boolean value to indicate whether the triangle is filled. The program should create a Triangle object with these sides and set the color and filled properties using the input. The program should display the area, perimeter, color, and true or false to indicate whether it is filled or not.
- In a triangle, the sum of any two sides is greater than the other side. The Triangle class must adhere to this rule. Create the IllegalTriangleException class, and modify the constructor of the Triangle class to throw an IllegalTriangleException object if a triangle is created with sides that violate the rule

East West University
Department of Computer Science and Engineering
A/2, Jahurul Islam Avenue, Jahurul Islam City, Aftabnagar, Dhaka

Lab Manual: 08

Lab Topic: File Handling

Course Code: CSE110 (Object Oriented Programming)

Course Instructor: Tanni Mittra, Senior Lecturer, CSE

Lab Objective

1. **Learn a** mechanism to handle File in Java program

Lab Activities:

Lab Problem 1:

Write a program to create a file named Lab08_01.txt if it does not exist. Append new data to it if it already exists. Write 100 integers created randomly into the file using text I/O. Integers are separated by a space.

Lab Problem 2:

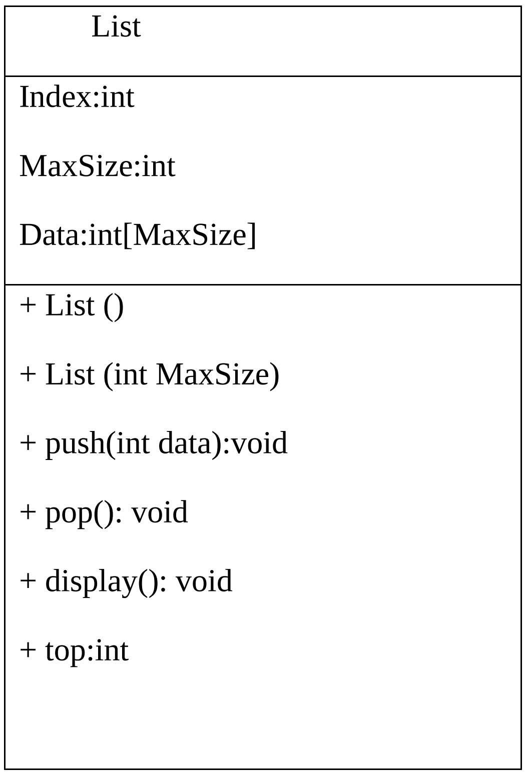
Write a program to create a file named Lab08_02.dat if it does not exist. Append new data to it if it already exists. Write 100 integers created randomly into the file using binary I/O. Integers are separated by a space.

Lab Problem 3:

Write a program that reads lines of characters from a text file and writes each line as a UTF-8 string into a binary file.

Lab Problem 4:

Consider the following class diagram and convert the class diagram into corresponding Java code.



a. Inside the no argument constructor set MaxSize attributes to 10 and initialize Index attributes to -1. Inside the second constructor set MaxSize variables by user provided value and also initialize Index attributes to -1.
b. Inside the push method add a new item in the Data array and the index of the array will be handled by Index instance variables. Each time a new data is added and Index attributes will be incremented by 1.
c. The pop method will remove one element of the array from last. Each time an element removed from element Index attribute will be decreased by one.
d. The top method will return an element which is added at last in the list.
e. The display method will display the list of the Data array

Write a program that creates five List objects and stores them in a file named Lab08_04.dat