CS102

Practice One

Review of Object-Oriented Programming

Step.0 Read the following introduction

Introduction

The object-oriented programming paradigm can enable you to develop large scale software effectively.

Object oriented programming is a paradigm; it is a mindset that a programmer can have to solve problems in developing software using object and classes.

An object represents an entity in the real world that can be uniquely identified. For instance, a student, a desk, a circle, a button, an account can all be viewed as objects.

An object has an identity, state and behavior.

The state of an object encapsulates the properties of an object. It represents data fields and their current values (known as state). For instance, consider a student object that has a data fields student ID number, student name, and student email address that characterize uniquely the student object.

The behavior of an object it often referred as actions refers to the methods defined in the object.

Every object can have a behavior which represented by its methods. For instance, you may define methods for the student object named getEmailAddress() that can return the email address of the student. A student object can also have a method setEmailAddress() that student object can invoke to change the email address of the student object.

Objects of the same type are defined using a common class

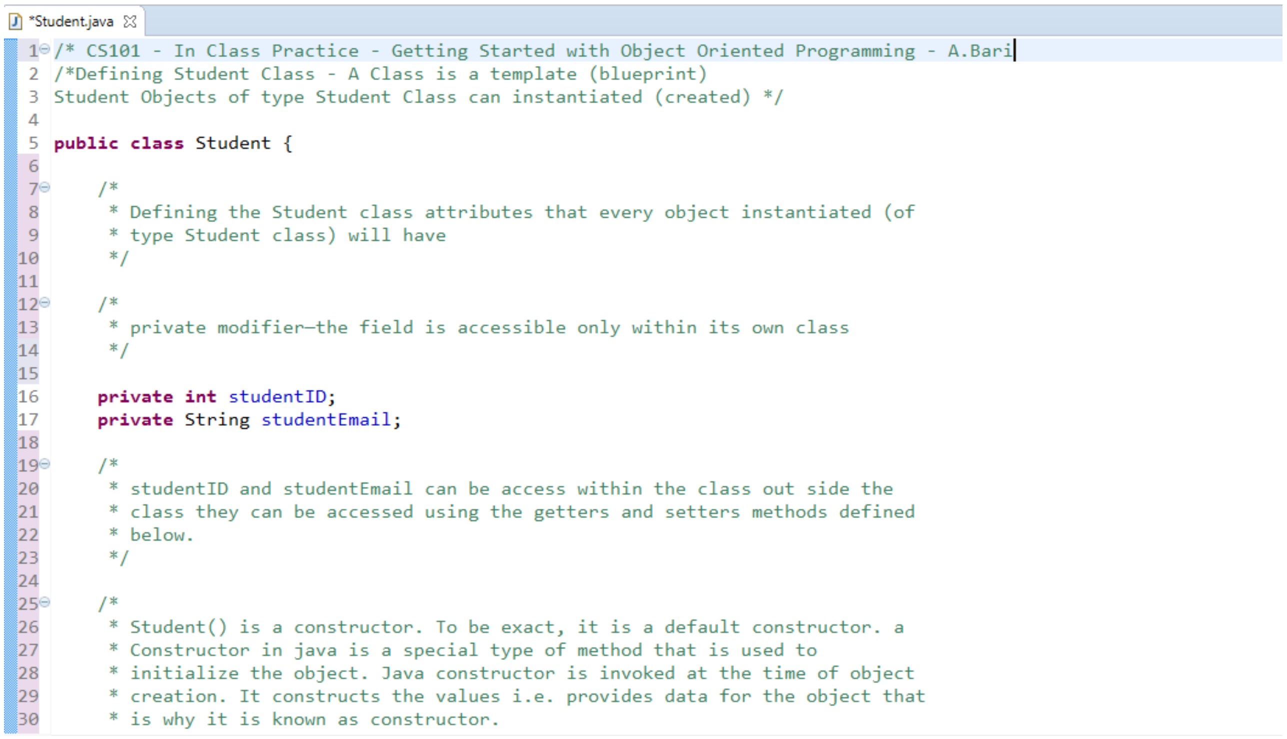
- A class is a template, blue print, or contract that defines what an object's data fields and methods will be.
- An object is an instance of a class
- Many objects of the same class can be created
- Creating an instance of class is referred to as instantiation
- The relationship between classes and objects are analogous to cookie-cutter and cookie
- Every class you define should have a special method named constructor, this method should have the same name as the class and it will allow you to create an object out of the defined class

Example:

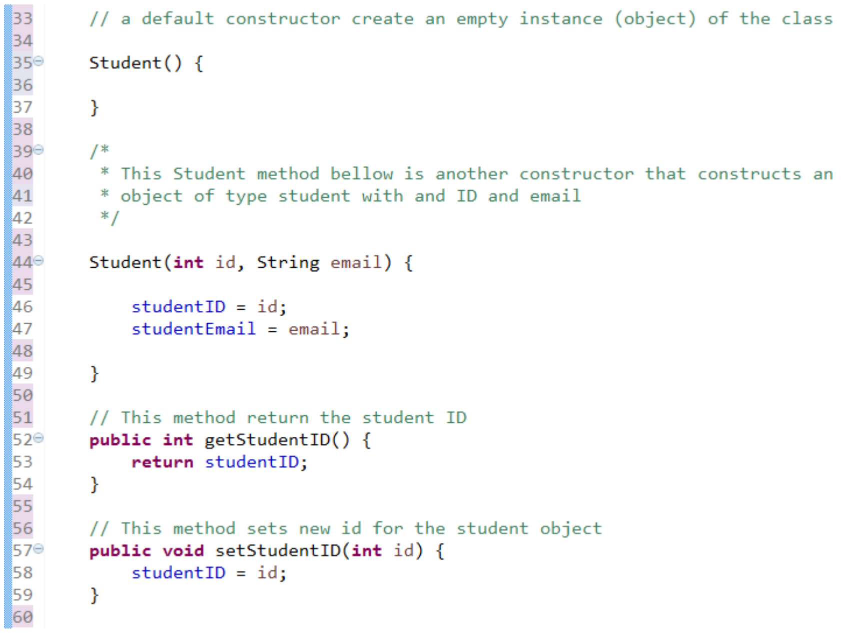
1. Open Eclipse
2. Create a new project
3. Create a new class named Student
4. Type the following into you Student class:

Step.1 Create a Student class and instantiate objects of the Student class

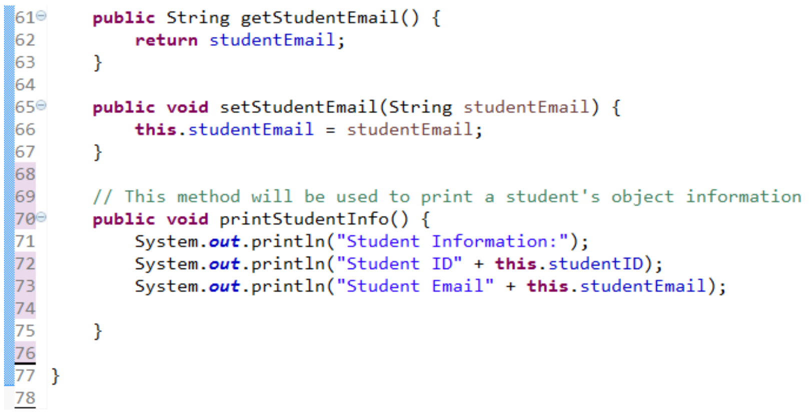
You do not have to type the comments



```java
/* CS101 - In Class Practice - Getting Started with Object Oriented Programming - A.Bari
/*Defining Student Class - A Class is a template (blueprint)
Student Objects of type Student Class can instantiated (created) */

public class Student {

    /*
     * Defining the Student class attributes that every object instantiated (of
     * type Student class) will have
     */

    /*
     * private modifier—the field is accessible only within its own class
     */

    private int studentID;
    private String studentEmail;

    /*
     * studentID and studentEmail can be access within the class out side the
     * class they can be accessed using the getters and setters methods defined
     * below.
     */

    /*
     * Student() is a constructor. To be exact, it is a default constructor. a
     * Constructor in java is a special type of method that is used to
     * initialize the object. Java constructor is invoked at the time of object
     * creation. It constructs the values i.e. provides data for the object that
     * is why it is known as constructor.
```



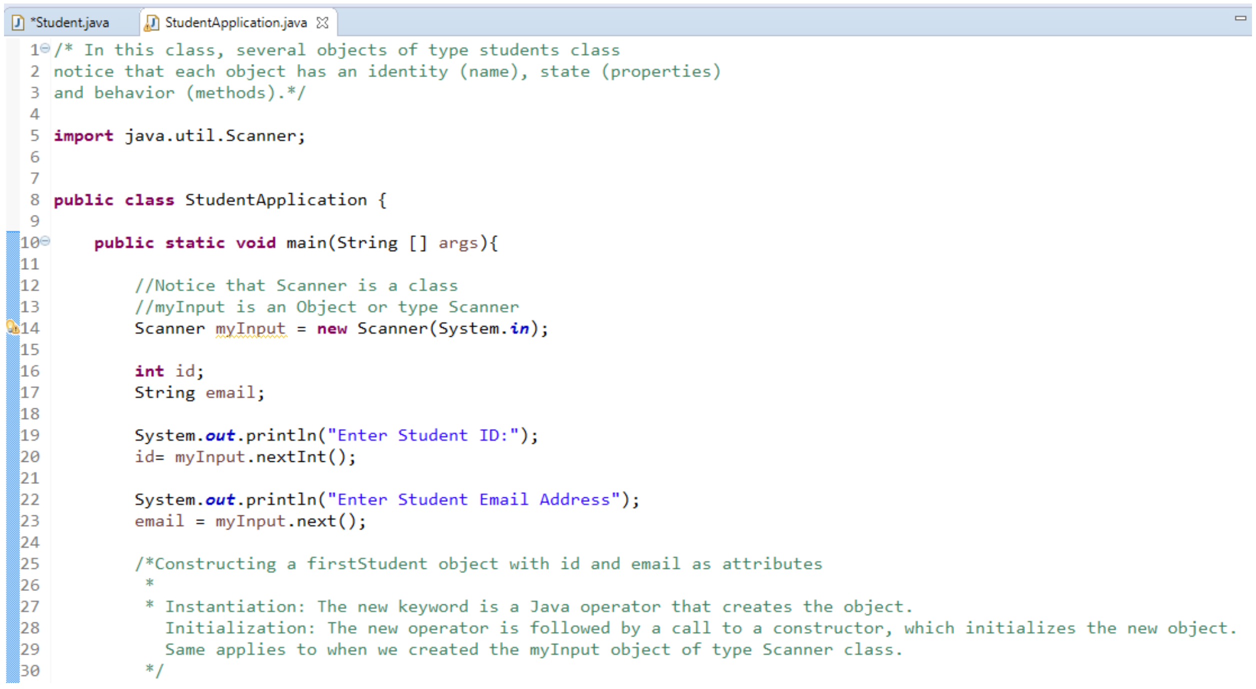
```java
    // a default constructor create an empty instance (object) of the class

    Student() {

    }

    /*
     * This Student method bellow is another constructor that constructs an
     * object of type student with and ID and email
     */

    Student(int id, String email) {

        studentID = id;
        studentEmail = email;

    }

    // This method return the student ID
    public int getStudentID() {
        return studentID;
    }

    // This method sets new id for the student object
    public void setStudentID(int id) {
        studentID = id;
    }

```



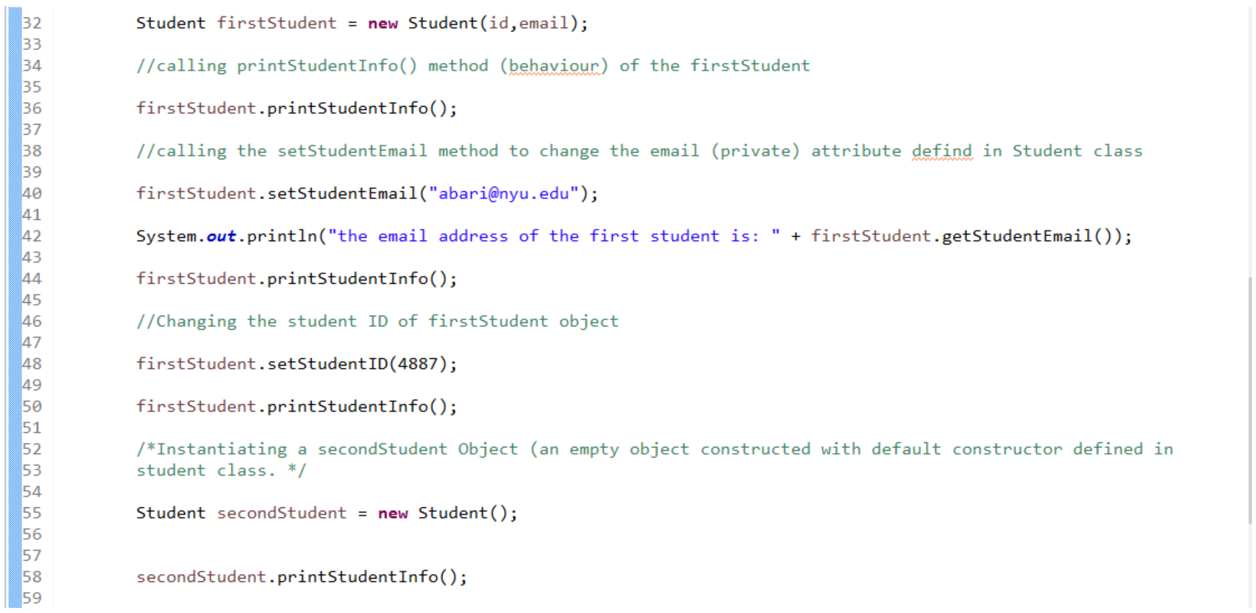
```java
    public String getStudentEmail() {
        return studentEmail;
    }

    public void setStudentEmail(String studentEmail) {
        this.studentEmail = studentEmail;
    }

    // This method will be used to print a student's object information
    public void printStudentInfo() {
        System.out.println("Student Information:");
        System.out.println("Student ID" + this.studentID);
        System.out.println("Student Email" + this.studentEmail);

    }

}

```

Step.2 Create the **StudentApplication** Class that will be used to create **Objects** of type Student

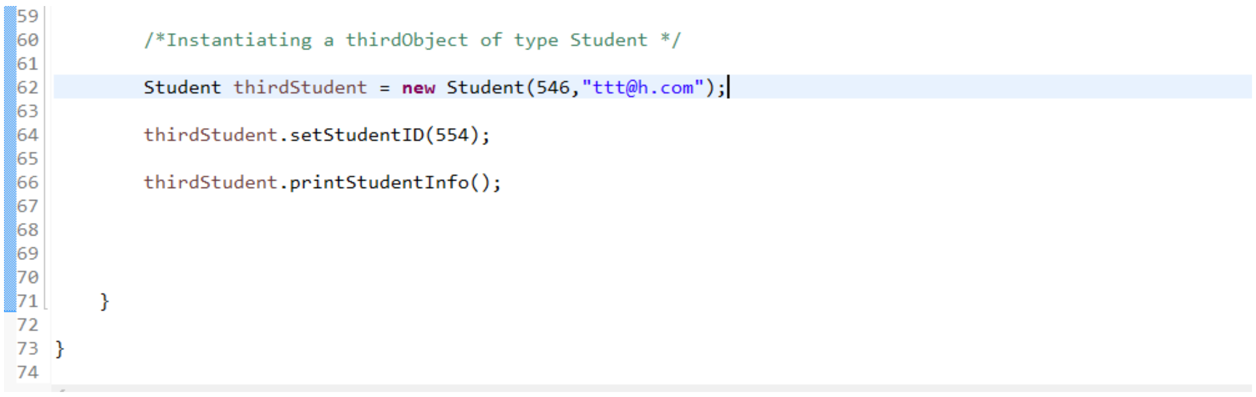
- Under the same project create class called StudentApplication
- Type the following code and please read the comments
- Notice the use of the getters and setters methods



```java
/* In this class, several objects of type students class
notice that each object has an identity (name), state (properties)
and behavior (methods).*/

import java.util.Scanner;


public class StudentApplication {

    public static void main(String [] args){

        //Notice that Scanner is a class
        //myInput is an Object or type Scanner
        Scanner myInput = new Scanner(System.in);

        int id;
        String email;

        System.out.println("Enter Student ID:");
        id= myInput.nextInt();

        System.out.println("Enter Student Email Address");
        email = myInput.next();

        /*Constructing a firstStudent object with id and email as attributes
         *
         * Instantiation: The new keyword is a Java operator that creates the object.
           Initialization: The new operator is followed by a call to a constructor, which initializes the new object.
           Same applies to when we created the myInput object of type Scanner class.
         */
```



```java
        Student firstStudent = new Student(id,email);

        //calling printStudentInfo() method (behaviour) of the firstStudent

        firstStudent.printStudentInfo();

        //calling the setStudentEmail method to change the email (private) attribute defind in Student class

        firstStudent.setStudentEmail("abari@nyu.edu");

        System.out.println("the email address of the first student is: " + firstStudent.getStudentEmail());

        firstStudent.printStudentInfo();

        //Changing the student ID of firstStudent object

        firstStudent.setStudentID(4887);

        firstStudent.printStudentInfo();

        /*Instantiating a secondStudent Object (an empty object constructed with default constructor defined in
        student class. */

        Student secondStudent = new Student();


        secondStudent.printStudentInfo();

```



```
        /*Instantiating a thirdObject of type Student */

        Student thirdStudent = new Student(546,"ttt@h.com");

        thirdStudent.setStudentID(554);

        thirdStudent.printStudentInfo();



    }

}

```

Step 3:

- Modify the Student class to also include **the student first name and student last name.**
- Create a new class called StudentsList that has **an array of five student objects (an array of type Student).**
- Populate the five students' objects from the user using the Scanner.
- Iterate through the objects and print them to the screen using the printStudentInfo method defined in the Student class (you can create a method called PrintAll)
- Create method called ***SearchStudent*** that takes a student id and returns a Student Object.
- found otherwise you can print that the student of the given id is not found.
- Create a method called EditStudent that will edit a student first name and last name.
- Create a method called DeleteStudent that will delete a student using it is id.
- Create a method called AddStudent that will add a student to the array.
- In the studentApplication class instantiate an object of type studentlist and create a menu to add, delete, search, edit a student.

Step 4.

Re-do Step 3 using an ArrayList instead of an array and add a method that will sort the students by ID.