- Method overloading

```java
Course.java   User.java   Admin.java   Student.java

    public void viewAllCourses(ArrayList<Course> list) {
        super.viewAllCourses(list);
    }
```

```java
Course.java   User.java   Admin.java   Student.java   CourseRegistrationSystem.java   InterfaceAdmin.java

        }
    }

    public void viewAllCourses(String firstName, String lastName, ArrayList<Course> list) {

        System.out.println("Courses registered by " + firstName + lastName + ": ");
        for(int i=0; i<allStudent.size(); i++) {
            if(allStudent.get(i).getFirstName().equalsIgnoreCase(firstName) && allStudent.get(i).getLastName().equalsIgnoreCase(lastName)) {
                System.out.println(allStudent.get(i).getMyCourse());
            }
        }
    }
```

In Class Admin, there are two Method viewAllCourses with the same name but different argument types.

- Method overriding (at least two examples)

```java
Course.java   User.java   Admin.java   Student.java   CourseRegistrationSystem.java   InterfaceAdmin.java

public void viewAllCourses(ArrayList<Course> list) {
    for(int i=0; i<list.size(); i++) {
        System.out.println(list.get(i).getCourseName() + ", " + list.get(i).getCourseId() + ", " + list.get(i).getMaximumStudents() + ", "
        + list.get(i).getCourseInstructor() + ", " + list.get(i).getCourseSectionNumber() + ", " + list.get(i).getCourseLocation());
    }
}

public void viewAllFullCourses(ArrayList<Course> list) {
    int count = 0;

    for(int i=0; i<list.size(); i++) {
        if(list.get(i).getMaximumStudents() == list.get(i).getCurrentStudents()) {
            System.out.println("All full courses are: " + list.get(i).getCourseName() + ", " + list.get(i).getCourseId() + ", " + list.get(i).getMaximumS
                    + list.get(i).getCourseInstructor() + ", " + list.get(i).getCourseSectionNumber() + ", " + list.get(i).getCourseLocation());
            count++;
        }
    }
    if(count == 0)
        System.out.println("There are no courses that are full.");
}
```

```java
Course.java   User.java   Admin.java   Student.java   CourseRegistrationSystem.java   InterfaceAdmin.java

@Override
public void viewAllCourses(ArrayList<Course> list) {
    for (int i=0; i<list.size(); i++) {
        System.out.println("All courses: " + list.get(i).getCourseName());
    }
}

@Override
public void viewAllFullCourses(ArrayList<Course> list) {
    int count = 0;

    for(int i=0; i<list.size(); i++) {
        if(list.get(i).getMaximumStudents() != list.get(i).getCurrentStudents()) {
            System.out.println("NOT full courses are: " + list.get(i).getCourseName() + ", " + list.get(i).getCourseId() + ", " + list.get(i).getMaximum
                    + list.get(i).getCourseInstructor() + ", " + list.get(i).getCourseSectionNumber() + ", " + list.get(i).getCourseLocation());
            count++;
        }
    }
    if(count == 0)
        System.out.println("All courses are full.");
}
```

Method viewAllCourses and Method viewAllFullCourses are overridden. In both examples, subclass (Student) overrides superclass (User), and only the subclass' method implementation is used.

- Abstract Class

```java
import java.util.*;

public abstract class User {
    protected String username;
    protected String password;
    protected String firstName;
    protected String lastName;
    protected ArrayList <Student> studentListOfAClass = new ArrayList<Student>();
    protected static ArrayList <Student> allStudent = new ArrayList<Student>();
```

Class User is an abstract class.

- Inheritance

```java
import java.util.*;

public class Admin extends User implements InterfaceAdmin{
    Scanner input = new Scanner(System.in);

import java.util.ArrayList;

public class Student extends User implements InterfaceStudent{
```

Class Admin and Class Student extends Class User.

- Polymorphism

```java
public void viewAllCourses(ArrayList<Course> list) {
    for(int i=0; i<list.size(); i++) {
        System.out.println(list.get(i).getCourseName() + ", " + list.get(i).getCourseId() + ", " + list.get(i).getMaximumStudents() + ", "
        + list.get(i).getCourseInstructor() + ", " + list.get(i).getCourseSectionNumber() + ", " + list.get(i).getCourseLocation());
    }
}

public void viewAllFullCourses(ArrayList<Course> list) {
    int count = 0;

    for(int i=0; i<list.size(); i++) {
        if(list.get(i).getMaximumStudents() == list.get(i).getCurrentStudents()) {
            System.out.println("All full courses are: " + list.get(i).getCourseName() + ", " + list.get(i).getCourseId() + ", " + list.get(i).getMaximumS
                    + list.get(i).getCourseInstructor() + ", " + list.get(i).getCourseSectionNumber() + ", " + list.get(i).getCourseLocation());
            count++;
        }
    }
    if(count == 0)
        System.out.println("There are no courses that are full.");
}
```

```java
@Override
public void viewAllCourses(ArrayList<Course> list) {
    for (int i=0; i<list.size(); i++) {
        System.out.println("All courses: " + list.get(i).getCourseName());
    }
}

@Override
public void viewAllFullCourses(ArrayList<Course> list) {
    int count = 0;

    for(int i=0; i<list.size(); i++) {
        if(list.get(i).getMaximumStudents() != list.get(i).getCurrentStudents()) {
            System.out.println("NOT full courses are: " + list.get(i).getCourseName() + ", " + list.get(i).getCourseId() + ", " + list.get(i).getMaximum
                    + list.get(i).getCourseInstructor() + ", " + list.get(i).getCourseSectionNumber() + ", " + list.get(i).getCourseLocation());
            count++;
        }
    }
    if(count == 0)
        System.out.println("All courses are full.");
}
```

Method viewAllCourses and Method viewAllFullCourses are overridden. In both examples, subclass (Student) overrides superclass (User), and only the subclass' method implementation is used.

Method overriding is a form of polymorphism.

- Encapsulation

```
Course.java   User.java   Admin.java   Student.java   CourseRegistration
import java.util.*;

public class Course implements java.io.Serializable{
    private String courseName;
    private String courseId;
    private int maximumStudents;
    private int currentStudents;
    private ArrayList<Student> listOfNames = new ArrayList<Student>();
    private String courseInstructor;
    private int courseSectionNumber;
    private String courseLocation;
```

In Class Course, all the variables are declared private. Class Course interacts with other class through constructor, getters, setters, and other public method. This way, we can hide information and data.

- The concept of ADT (Abstract Data Types)

ADT is a mathematical description of a data structure. Each operation is described by an ADT, but not how it is carried out. There are many int, String, List (ArrayList) used in the code which are all ADT.

```
Course.java   User.java   Admin.java   Student.java   CourseRegistration
import java.util.*;

public class Course implements java.io.Serializable{
    private String courseName;
    private String courseId;
    private int maximumStudents;
    private int currentStudents;
    private ArrayList<Student> listOfNames = new ArrayList<Student>();
    private String courseInstructor;
    private int courseSectionNumber;
    private String courseLocation;
```