Teng Liang
Examples of concept

▪ Method overloading

Used method overloading in constructors. This help when specifying which constructor to call with which parameters.

For example: `public Student(){`

```
        }

        public Student(String username, String password, String first_name, String last_name) {
    super(username, password, first_name, last_name);

}

public Student(String first_name, String last_name) {
    super(first_name, last_name);
}
```

▪ Method overriding (at least two examples)

Used a lot of methods that have multiple usages. This allows for simplicity and method reusability.

For example:

```
public String getPassword() {
                return password;
        }
```

This get password is used both by admin and student

▪ Abstract Class

Used abstract class for User. It utilizes methods for inherited classes and also contains members that will be used for the childclass.

For example: **abstract class** User **implements** Serializable{

```
// this user class has basic traits such as username, password, first and last name. Admin and student inherits from this class and uses these members through getters and setters.
    /**
             *
             */
// all username, password, and names are private members
             private static final long serialVersionUID = 1L;
             private String username;
             private String password;
             private String first_name;
             private String last_name;
// below are constructors for User class

             public User() {
                }

    public User(String first_name, String last_name) {
                        this.first_name = first_name;
                        this.last_name = last_name;
             }

    public User(String username, String password, String first_name, String last_name) {
        this.username = username;
        this.password = password;
        this.first_name = last_name;
        this.last_name = last_name;
```

▪ Inheritance

Used inheritance to inherit student and admin class from user class. This leads to it being able to user password, username, and all kinds of information from the user class.

For example:

```
public class Admin extends User implements Admin_methods {
             // the Admin class used by Admins and uses admin method interface available only to admin

    /**
             *
             */
             private static final long serialVersionUID = 1L;
             static ArrayList<Student> student_list = new ArrayList<Student>();
    static ArrayList<Course> course_list = new ArrayList<Course>();
```

```
// constructors below
        public Admin() {
                super();
                this.setUsername("admin");
                this.setPassword("admin001");
        }
```

▪ Polymorphism

Polymorphism is used everywhere. For instance multiple constructors I mentioned above is a use of polymorphism.

For example:

```
public Student(){


        }

        public Student(String username, String password, String first_name, String last_name) {
    super(username, password, first_name, last_name);

}

public Student(String first_name, String last_name) {
    super(first_name, last_name);
}
```

▪ Encapsulation:

Encapsulation is used to make sure private members aren't publicly accessible.

For example:

```
public class Course implements Serializable{

        // this class is the course class, contains all the members for the courses which are heaivly used in the
whole project

/**
         *
         */
        // private members for the courses
        private static final long serialVersionUID = 1L;
        private String course_name;
private String course_id;
        private int max_students;
```

```
private int registered_students;
private ArrayList<Student> student_list;
private String instructor_name;
private int section_number;
private String location;
// below are constructors for the course class
public Course() {

}
```