1. Method Overloading:

```
FileOperation.serialize();
FileOperation.serialize(studentsList);
```

   - Method overloading refers to the idea that methods can have the same name but different variables. When serializing, I used method overloading since we need to take in students list when serializing for student, and courses list when serializing courses. Java recognizes these methods as different.

2. Method Overriding:

```
admin.adminMenu();
```

```
public void adminMenu()
{
    System.out.println("Would you like to: \n"
            + "1. Create a new course; \n"
            + "2. Delete a course; \n"
            + "3. Edit a course; \n"
            + "4. Display information for a given course; \n"
            + "5. Register a student; \n"
            + "6. View all courses; \n"
            + "7. View all courses that are full; \n"
            + "8. Write to a file all full courses; \n"
            + "9. View the students being registered in a course; \n"
            + "10. View the list of courses that a given student is being
            + "11. Sort courses \n"
            + "12. Exit \n"
            + "Please input the number of your choice!");
}
```

```
else if(choice==6)
{
    System.out.println("View all courses");
    admin.viewAllCourses();
}
```

   - Since Student and Admin classes are inherited from User, a lot of its methods are overridden. For example, the getter and setter methods used super keywords to override from their parent class User. A simple example in my program would be the admin menu and viewAllCourses method that allowed Admin access because it is inherited from the User.

3. Abstract Class

```
public abstract class User implements Serializable{
```

   - The User class could be an abstract class. It is a parent class of Student and Admin, and it does not need any instance of User.

4. Inheritance

```
public class Admin extends User implements AdminInterface, Serializable{
```

```
public class Student extends User implements StudentInterface, Serializable{
```

- Student and Admin classes are inherited from the User class and their methods are largely inherited. Aside from the setter and getter methods, the method viewCourses is used by Admin and inherited from User. Admin has access to all methods in User.

5. Polymorphism

```
public class Admin extends User implements AdminInterface, Serializable{
```

```
public class Student extends User implements StudentInterface, Serializable{
```

- Polymorphism refers to how objects take many forms. Therefore inheritance will already be a great example of polymorphism. Also, the admin and student class implement their own interface and serializable.

6. Encapsulation

```
public class FileOperation implements Serializable{
    /*
     * This class is used to serialize, deserialize and read files
     */
    private static ArrayList<Course> courseList ;
    private static ArrayList<Student> studentList ;
    //private FileOperation f=new FileOperation();
    public static ArrayList<Course> getCourseList()
    {
        return courseList;
    }
    public static ArrayList<Student> getStudentList()
    {
        return studentList;
    }
```

- Most fields in FileOperation, and other classes such as course, student, and admin are all private with getter methods to keep it safe.

7. ADT

```
public abstract class User implements Serializable{
```

- I used ArrayList to store students and course, and there are classes that store the templates of each object: students, admin, and courses.