**Java Software Development**

**Laboratory III**

*The purpose of this laboratory is to practise File and I/O Streams and Comparator*

Start from Employee project from basic Inheritance sample code.

1- Study the Employee polymorphism example. Reuse your Address and postcode classes from lab I and II and add an Address field to the Employee and add toString() method to parent and children classes.

2- Study InheritanceFiling.java from file sample code to learn how to read and write parent and children objects in the text file. Create a list of Employees and add several SalaryEmployee, and CasualEmployee objects to the list. Write the list in the file, clear the list, read it again from the file, and print it. Note that you need extra fields for the Address and PostCode in each record.

3- Study InheritanceFilingSerialization.java from file sample code to learn how to read and write parent and children objects in the binary file by using java Serialization. Write the Employee list in the binary file, clear the list, read it again from the file, and print it.

4- You already added address and postcode objects to your employees and know how to work with 1:1 aggregation or composition (has a) relationship. In this stage you need to work with 1:n aggregation/composition. Study InheritanceFilingSerialization2.java from file sample code to learn how to read and write classes with complex aggregation/composition in the binary file by using java Serialization. Add a list of Projects to the Employee class. The Project has an ID and a name. Modify toString() method to convert the list of the projects to string in addition to the Employee details. Use the Employee constructor to set the list. Write the new Employee list in the binary file, clear the list, read it again from the file, and print it.

5- For the new Employee with the list of the Projects extend your code from step 2 to write and read the list of Employees in/from the text file.

6- Write the clone() for the Employee. Note that you need to clone all the projects in the Project list. Create a shallow copy and deep copy of the Employee list in your test code. Add comapreTo() to Employee to compare the list based on Employee name. Add a Comparator to compare two Employees based on postcode. In your main sort the original list by using Comparable and the shallow copy list by using Comparator