Lab 8: Abstract Classes and Interfaces

Objectives

- To understand the concept of abstract classes.
- To be able to differentiate between abstract and concrete classes.
- To define a natural order using the `Comparable` interface.
- To know the similarities and differences between an abstract class and an interface.
- To declare custom interfaces

Theory

Abstract classes are used to declare common characteristics of subclasses. An abstract class cannot be instantiated. It can only be used as a superclass for other classes that extend the abstract class. Abstract classes are declared with the abstract keyword. Abstract classes are used to provide a template or design for concrete subclasses down the inheritance tree.

An abstract method is a method signature without implementation. Its implementation is provided by the subclasses. A class that contains abstract methods **must** be declared abstract.

An interface defines one or more constant identifiers and abstract methods. A separate class *implements* the interface class and provides the definition of the abstract methods. Interfaces are used as a design technique to help organize properties (identifiers) and behaviors (methods) the implementing classes may assume.

Syntax

Abstract Class

```
[modifier] abstract class Class-Name {
  /** Body of the class */
}
```

Abstract Method

```
[modifier] abstract data-type method-name([parameters-declarations]) {…}
```

Interface

```
[modifier] interface InterfaceName {
  /** Constant declarations */
  /** Method signatures */
}
```

Exercises

1. Write a method that returns the largest object in an array of objects. The method signature is:
`public static Object max(Object[] a).`
All the objects are instances of the `Comparable` interface. The order of the objects in the array is determined using the `compareTo` method.

Modify exercise 1 in Lab 7 in which the **`Account`** class will implement the **`compareTo()`** method in the **`comparable`** interface. This method will compare the amount of two accounts. Write a test program that creates an array of ten `Accounts`, and finds the largest `Account` in the array.

2. Modify exercise 2 in Lab 7 to make the `Employee` class an abstract class and the `earning()` method as an abstract method. Also allow each of the Employee subclasses implement the `compareTo()` method in the comparable interface. The `compareTo` method compares between the employees earnings.

Write a method that finds the total earning of all the employees in an array. The method signature is:

```
public static double totalEarning(Employee[] a)
```

Write another method that sort an array of Objects using the `compareTo` method. The signature of this method is:

```
public static void sort(Object[] a)
```

Write a test program that creates an array of employees and computes their total earnings using the `totalEarning` method. Also create an array of basic salary employees then sort the array using the `sort` method.