Lab 1: Inheritance/Polymorphism

Exercise 1:

- Create a class called Employee whose objects are records for an employee. This class will be a derived class of the class Person which you will have to copy into a file of your own and compile.
- An employee record has an employee's name (inherited from the class Person), an annual salary represented as a single value of type double, a year the employee started work as a single value of type int and a national insurance number, which is a value of type String.
- Your class should have a reasonable number of constructors and accessor methods, as well as an equal's method. Write another class containing a main method to fully test your class definition.

Exercise 2:

- Create a Dog class. Class Labrador and Yorkshire contain declarations for classes that extend Dog. Class DogTest.java contains a simple driver program that creates a dog and makes it speak.
- Add statements in DogTest after you create and print the dog to create and print a Yorkshire and a Labrador. Note that the Labrador constructor takes two parameters: the name and color of the labrador, both strings.
- Add code to DogTest class to print the average breed weight for both your Labrador and your Yorkshire. Use the avgBreedWeight () method for both.
- Add an abstract int avgBreedWeight() method to the Dog class. Remember that this means that the word abstract appears in the method header after public, and that the method does not have a body (just a semicolon after the parameter list). It makes sense for this to be abstract, since Dog has no idea what breed

it is. Now any subclass of Dog must have an avgBreedWeight method; since both Yorkshire and Labrador do, you should be all set.

Exercise 3:

- Using the Account class as a base class, write two derived classes called SavingsAccount and CurrentAccount.
  A SavingsAccount object, in addition to the attributes of an Account object, should have an interest variable and a method which adds interest to the account.
  A CurrentAccount object, in addition to the attributes of an Account object, should have an overdraft limit variable.
  Ensure that you have overridden methods of the Account class as necessary in both derived classes.
- Now create a Bank class, an object of which contains an array of Account objects. Accounts in the array could be instances of the Account class, the SavingsAccount class, or the CurrentAccount class. Create some test accounts (some of each type).
- Write an update method in the bank class. It iterates through each account, updating it in the following ways: Savings accounts get interest added (via the method you already wrote); CurrentAccounts get a letter sent if they are in overdraft.
- The Bank class requires methods for opening and closing accounts, and for paying a dividend into each account.

Hints:

- Note that the balance of an account may only be modified through the deposit(double) and withdraw(double) methods.
- The Account class should not need to be modified at all.
- Be sure to test what you have done after each step.

Exercise 4:
Create a class 'Student' with three data members which are name, age and address. The constructor of the class assigns default values name as "unknown", age as '0' and address as "not available". It has two members with the same name 'setInfo'. First method has two parameters for name and age and assigns the same whereas the second method takes has three parameters which are assigned to name, age and address respectively. Print the name, age and address of 10 students.
Hint - Use array of objects

Exercise 5 :
Write a program in Test Driven approach to create interface named Calculator. In this interface the member function is square. Implement this interface in arithmetic class. Create one new class called CalculatorApp in this class use the object of arithmetic class.

Exercise 6:
A User has has deposited money 1000, 1500 and 2000 in banks-Bank SBI, Bank Hdfc and Bank Kotak respectively. We have to print the money deposited by User in a particular bank. Create a class 'Bank' with a method 'getBalance' which returns 0. Make its three subclasses named 'Sbi', 'Hdfc' and 'Kotak' with a method with the same name 'getBalance' which returns the amount deposited in that particular bank. Call the method 'getBalance' by the object of each of the three banks.