FRESHER ACADEMY

JAVA BASICS

Contents

Objective 2
Business needs 2
Working requirements 2
Product architecture 2
Technologies 2
Stored Data 2

Objective

- Understand and practise with Classes, Object, Access Modifier, Constructors, Is A Super, this keyword.
- Understand and practise with Control-of-flow.

Business needs

- TBD

Working requirements

Working environment: Eclipse IDE.

Delivery: Source code, deployment and testing, reviewing evident packaged in a compress archive.

Product architecture

- N/A

Technologies

The product implements one or more technology:

- Java basics
- Control of Flows
- OOP

Stored Data

- N/A

Project Descriptions

1. **Exercise 1**

   Create a class called **Book** to represent a book. A Book should include four pieces of information as:

   - instance variables-a book name,
   - an ISBN number,
   - an author name and a publisher.

Your class should have a constructor that initializes the four instance variables. Provide a **mutator** method and **accessor** method (query method) for each instance variable.

In addition, provide a method named **getBookInfo** that returns the description of the book as a String (the description should include all the information about the book). You should use *this* keyword in member methods and constructor.

Write a test application named **BookTest** to create an array of object for 30 elements for class Book to demonstrate the class Book's capabilities.

Estimated time: 30 mins

2. **Exercise 2**

Create a class called **Employee** that includes three pieces of information as instance variables:

- a first name (type String),
- a last name (type String) and
- a monthly salary (double).

Your class should have a constructor that initializes the three instance variables. Provide a set and a get method for each instance variable. If the monthly salary is not positive, set it to 0.0.

Write a test application named EmployeeTest that demonstrates class Employee's capabilities. Create two Employee objects and display each object's yearly salary. Then give each Employee a 10% raise and display each Employee's yearly salary again.

Estimated time: 30 mins

3. **Exercise 3**

Create a super class called **Car**. The Car class has the following fields and methods.

- int speed;
- double regularPrice;
- String color;

- double getSalePrice();

Create a sub class of Car class and name it as **Truck**. The Truck class has the following fields and methods.

- int weight;
- double getSalePrice(); //If weight>2000,10% discount. Otherwise, 20% discount.

Create a subclass of Car class and name it as **Ford**. The Ford class has the following fields and methods.

- int year;
- int manufacturerDiscount;
- double getSalePrice();

  //From the sale price computed from Car class, subtract the manufacturer Discount

Create a subclass of Car class and name it as **Sedan**. The Sedan class has the following fields and methods.

- int length;
- double getSalePrice(); // If length > 20 feet, 5% discount. Otherwise, 10% discount.

Create **MyOwnAutoShop** class which contains the main() method. Perform the following within the main() method.

- Create an instance of **Sedan** class and initialize all the fields with appropriate values. Use super(...) method in the constructor for initializing the fields of the superclass.
- Create two instances of the **Ford** class and initialize all the fields with appropriate values. Use super(...) method in the constructor for initializing the fields of the super class.
- Create an instance of **Car** class and initialize all the fields with appropriate values.

Display the sale prices of all instance.

Estimated time: 30 mins

4. Exercise 4

Xây dựng lớp **"NumberList"** để mô tả một dãy số, gồm các phương thức sau:

- Phương thức "input" dùng để nhập dãy số từ bàn phím
- Phương thức "print" dùng để in dãy số ra màn hình
- Hàm tạo **NumberList** (int n) dùng để khởi tạo một mảng gồm n phần tử

b) Xây dựng giao diện Sort như sau:

```
public interface Sort {
```

```
    public void  sort();
}
```

c) Xây dựng các lớp **"QuickSort"**, **"SelectionSort"**, **"InsertSort"** bằng cách kế thừa từ lớp **NumberList** và triển khai giao diện Sort để thực hiệp việc sắp xếp: nhanh, chọn trực tiếp, chèn trực tiếp.

Estimated time: 90 mins

The End!