Classes

1 Introduction

Java is an object-oriented programming language, meaning it focuses on creating objects that contain both data and methods to manipulate that data. The main elements of object-oriented programming are classes, objects, methods, fields, and inheritance.

2 Classes

A class in Java serves as a blueprint that defines the properties and behavior of objects. It consists of fields (variables) and methods that manipulate these fields. Here's an example of a class definition:

```
public class Person {
    String name;
    int age;

    public void introduceYourself() {
        System.out.println("Hello, my name is " + name);
    }
}
```

3 Class Fields

Class fields (also called attributes or properties) are variables that store information about objects of a given class. In the example above, `name` and `age` are fields of the `Person` class.

4 Methods

Methods are functions defined within a class that perform specific operations on objects of that class. Each method has a signature, which includes the method's name, return type, and parameter list. An example of a method is `introduceYourself()`, which outputs a message to the console.

```
public void introduceYourself() {
    System.out.println("Hello, my name is " + name);
}
```

5 Access Modifiers

Access modifiers specify the level of accessibility for classes, methods, and fields. Java has the following access modifiers:

- `public` - accessible from any class,
- `protected` - accessible within the same package and by subclasses,
- `private` - accessible only within the same class,
- package-private (no modifier) - accessible only within the same package.

Example:

```
public class Person {
    private String name;
    protected int age;
}
```

6 Inheritance

Inheritance allows you to create a new class based on an existing class, so the new class inherits the fields and methods of the base class. The `extends` keyword is used to indicate that a class is derived from another class.

Example:

```
public class Student extends Person {
    int studentId;

    public void introduceYourself() {
        System.out.println("Hello, I am a student, my name is " + name);
    }
}
```

The `Student` class inherits fields and methods from the `Person` class but also has its own field `studentId` and an overridden `introduceYourself()` method.

7 Exercises

To practice and reinforce these concepts, try solving the following exercises:

1. Define a `Car` class with fields for `make` (String), `model` (String), and `year` (int). Add a method `displayInfo()` that prints out the details of the car.

2. Modify the `Car` class so that `make` and `model` are `private` fields. Add getter and setter methods to access and modify these fields from outside the class.

3. Create a new class `ElectricCar` that extends the `Car` class. Add an additional field `batteryCapacity` (int). Override the `displayInfo()` method to include information about the battery capacity.

4. In the `ElectricCar` class, add a second `displayInfo()` method that accepts a parameter for displaying the battery capacity in kilowatt-hours (kWh).

5. Write a main method to create an instance of `ElectricCar`, set its fields using the constructor and setter methods, and call `displayInfo()`.