Theory: Referencing subclass objects

22 minutes

Verify to skip | Start practicing

As you know, in Java, classes are organized into a hierarchy, which allows us to refer to objects in different ways. A class that is derived from another class is called a subclass. A class from which the subclass is derived is called a superclass. In this topic, you will learn the two ways to refer to a subclass object. You will find out when it is a good idea to use a superclass reference and what restrictions you should keep in mind.

1 required topic

✓ Protected modifier (In project)

4 dependent topics

✓ The Object class (In project)

Encapsulating object creation

Polymorphism

Runtime type checking

§1. How to refer to a subclass object

There are two ways to refer to a subclass object:

1. Using the subclass reference: you can use the subclass reference to refer to its object;

2. Using the superclass reference: you can use a reference variable of the superclass to refer to any subclass object derived from that superclass because a subclass is a special case of the superclass.

Let's consider an example of a class hierarchy.

```
class Person {

    protected String name;
    protected int yearOfBirth;
    protected String address;

    // public getters and setters for all fields
}

class Client extends Person {

    protected String contractNumber;
    protected boolean gold;

    // public getters and setters for all fields
}

class Employee extends Person {

    protected Date startDate;
    protected Long salary;

    // public getters and setters for all fields
}
```

As you know, each of the presented classes has a default no-args constructor.

Now let's see both approaches to the reference in action.

1. **Subclass reference**. We can create instances of the subclasses using the constructor:

```

Person person = new Person(); // the reference is Person, the object is Perso

Client client = new Client(); // the reference is Client, the object is Clien

Employee employee = new Employee(); // the reference is Employee, the object
```

In this case, we used **subclass references** because the types of the references and the created object are the same.

2. **Superclass reference**. When creating objects using the constructor, we can refer to a subclass object using the reference to the superclass:

```
Person client = new Client(); // the reference is Person, the object is Clien
Person employee = new Employee(); // the reference is Person, the object is E
```

In this case, we used the **superclass reference** because the references have the type of the superclass and the actual types of created objects are subclasses.

Remember, that:

- you cannot assign an object of one subclass to the reference of another subclass because they don't inherit each other:

```
Client whoIsIt = new Employee(); // it's impossible
```

- you cannot assign an object of the parent class to the reference of its subclass:

```
Client client = new Person(); // it's impossible too
```

> The basic rule goes like this:
> If class A is a superclass of class B and class B is a superclass of class C then a variable of class A can reference any object derived from that class (for instance, objects of the class B and the class C). This is possible because each subclass object is an object of its superclass but not vice versa.

§2. Accessing fields and methods through a superclass reference

We can use a superclass reference for any subclass object derived from it. However, we cannot access specific members of the subclass through the base class reference. We have access only to those members of the object that are defined by the type of reference.

Here is an example; in the considered hierarchy, each class has getters and setters to access protected fields from the outside.

```
Person employee = new Employee();

employee.setName("Ginger R. Lee"); // Ok
employee.setYearOfBirth(1980); // Ok
employee.setSalary(30000); // Compile-
time error, the base class "doesn't know" about the method
```

The superclass `Person` doesn't have the method `setSalary` of the class `Employee`. You cannot invoke the method through the superclass reference. The same rule holds for fields.

§3. Casting between superclass and subclass

You can always cast an object of a subclass to its superclass. It may also be possible to cast an object from a superclass type to a subclass, but only if the object is an instance of this subclass, otherwise a `ClassCastException` will be thrown. Be careful when casting a class to its subclass.

```
Person person = new Client();

Client clientAgain = (Client) person; // it's ok

Employee employee = (Employee) person; // the ClassCastException occurs here
```

After successfully casting a superclass to a subclass, we can access its members.

§4. When to use the superclass reference?

When to use a superclass reference in practice may not be so obvious. Moreover, using a superclass reference imposes some restrictions on accessing class members. There are two common cases:

- processing an array (or another collection) of objects which have different types from the same hierarchy;
- a method that accepts an object of the base class, but can also work with objects of its subclasses.

What we did is we combined both of these cases into a single example. Our method called `printNames` takes an array of `Person` and displays the names.

```
public static void printNames(Person[] persons) {
    for (Person person : persons) {
        System.out.println(person.getName());
    }
}
```

This method will work for an array with `Person`, `Client` and `Employee` objects.

```
Person person = new Employee();
person.setName("Ginger R. Lee");

Client client = new Client();
client.setName("Pauline E. Morgan");

Employee employee = new Employee();
employee.setName("Lawrence V. Jones");

Person[] persons = {person, client, employee};

printNames(persons);
```

The output is exactly as we expected:

```
Ginger R. Lee
Pauline E. Morgan
Lawrence V. Jones
```

As you can see, base class references have applications in some practical cases. Other cases of using the superclass references will be considered in topics related to **polymorphism**.

§5. Conclusion

You can refer to a subclass object in two ways, using the subclass or the superclass reference. A superclass reference can be used for any of its subclass objects but you cannot assign an object of the parent class to the reference of its subclass. Remember, that when referring to objects with a superclass reference you cannot invoke methods and fields of a subclass.

You can always cast an object of a subclass to a superclass — and vice versa, but only if the object is indeed an instance of the subclass.

Table of contents:

In practice, a superclass reference can be successfully applied when processing an array of objects which have the same parent class or when there is a method that accepts an object of the base class. You will learn about other cases of superclass references in topics related to polymorphism.

Report a typo

621 users liked this piece of theory. 51 didn't like it. **What about you?**

Start practicing

Verify to skip

Comments (38) Hints (0) Useful links (3) Show discussion