برنامه‌سازی پیشرفته

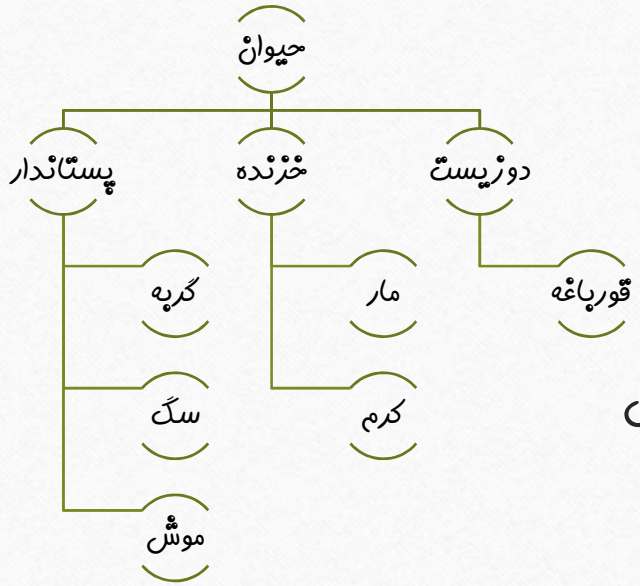
وراثت

مهدی مصطفی‌زاده

سرفصل مطالب

- مفهوم وراثت
- پیاده‌سازی وراثت در جاوا
- رابطه‌ی IS-A و پادالگوی «میراث مردود»
- Override
- super
- نمایش رابطه‌ی وراثت در نمودار کلاس UML
- وراثت و مقداردهی اولیه به اشیاء

مفهوم وراثت



- هر شئ از رده‌ی «مار»، شئ‌ای از رده‌ی «خزنده» و شئ‌ای از رده‌ی «حیوان» نیز است.
- هر شئ از رده‌ی «خزنده»، شئ‌ای از رده‌ی «حیوان» نیز است؛ اما نمی‌توان گفت که الزاماً شئ‌ای از رده‌ی «مار» هم هست.

پیاده‌سازی وراثت در جاوا

```java
package edu;

import family.Person;

public class Student extends Person {

}
```

```java
package main;

import edu.Student;

public class Main {

    public static void main(String[] args) {
        Student st = new Student();
        System.out.println(st.whatIsYourFatherName());
    }

}
```

- کلیدواژه‌ی extends
- متد whatIsYourFatherName() از رده‌ی پدر (Person) به رده‌ی فرزند (Student) به ارث رسیده است.
- سطح دسترسی protected

بازتعریف (Override) متدها

- رده‌ی فرزند، ویژگی‌ها و رفتارهای رده‌ی پدر را به ارث می‌برد.
- رده‌ی فرزند می‌تواند برخی از رفتارهای رده‌ی پدر را تغییر دهد (Override).
- رده‌ی فرزند نمی‌تواند ویژگی یا رفتار (متد) رده‌ی پدر را حذف نماید.

```
public class Student extends Person {

    @Override
    public String whatIsYourFatherName() {
        return getFather() == null ? "-" : "My father's name is " + getFather().getName();
    }

}
```

کلیدواژه‌ی super

- پیش از این، از کلیدواژه‌ی this برای دسترسی به اعضای شئ جاری و فراخوانی سازنده استفاده می کردیم.
- می توانیم از کلیدواژه‌ی super برای دسترسی به اعضای تعریف شده در رده‌ی پدر و فراخوانی سازنده‌ی رده‌ی پدر استفاده کنیم.

```
@Override
public String whatIsYourFatherName() {
    return getFather() == null ?
            super.whatIsYourFatherName() :
                "My father's name is " + getFather().getName();
}
```

وراثت و مقداردهی اولیه به اشیاء

```java
public class Person {

    protected String name;
    private int age;
    private Person father;

    public Person(String name, int age, Person father) {
        this.name = name;
        this.age = age;
        this.father = father;
    }

}
```

```java
public class Student extends Person {

    int studentId;

    public Student(String name, int age, Person father, int studentId) {
        super(name, age, father);
        this.studentId = studentId;
    }

}
```

```java
public class Student extends Person {

}
```

Implicit super constructor Person() is undefined for default constructor. Must define an explicit constructor

1 quick fix available:

رابطه‌ی IS-A

- رابطه‌ی IS-A (اصل جایگزینی Liskov):

Subclasses should not expect more or provide less than their superclass.

- «اسب» دارای دو رفتار «خوردن» و «راه رفتن» است.
- «عنکبوت» هم دارای دو رفتار «خوردن» و «راه رفتن» است؛ پس آن را فرزند «اسب» قرار می‌دهیم.
- بعد از مدتی، «یورتمه رفتن» نیز به رفتارهای اسب اضافه می‌شود.
- تصور کنید عنکبوتی را که یورتمه می‌رود!
- رده‌ی فرزند نمی‌تواند سطح دسترسی به یک متد را کاهش دهد.

ردەی Object

- ردەی Object، ردەی ریشە در سلسلە مراتب درختی وراثت بین ردەهاست.
- همەی ردەها بە شکل مستقیم یا غیرمستقیم زیرردەای از Object هستند.
- متدهای toString()، equals() و ...

toString **متد**

Student.java | Person.java | Main.java

```
1  package main;
2
3  import edu.Student;
4
5  public class Main {
6
7      public static void main(String[] args) {
8          Student st = new Student("Ali Ahmadi", 21, null, 123);
9          System.out.println(st);
```

```
<terminated> Hello [Java Application] C:\Program Files\Java\jdk1.8.0_112\bin\javaw.exe (Mar 7, 2020, 2:43:28 PM)
edu.Student@15db9742
```

Student.java | Person.java | Main.java

```
10          this.name = name;
11          this.age = age;
12          this.father = father;
13      }
14
15      @Override
16      public String toString() {
17          return name + " (" + age + ")";
18      }
```

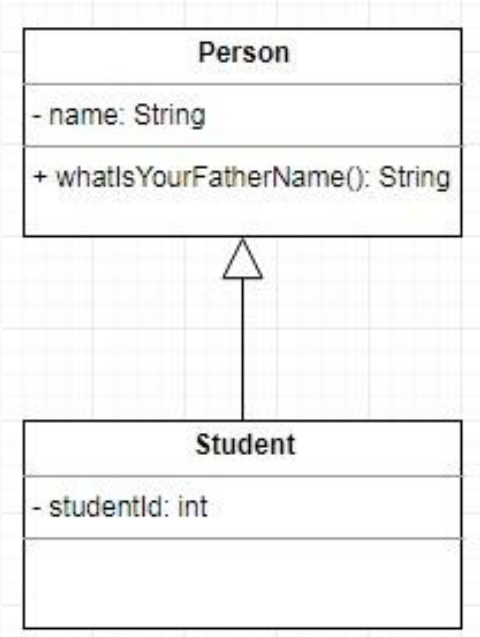
```
<terminated> Hello [Java Application] C:\Program Files\Java\jdk1.8.0_112\
Ali Ahmadi (21)
```

equals **متد**

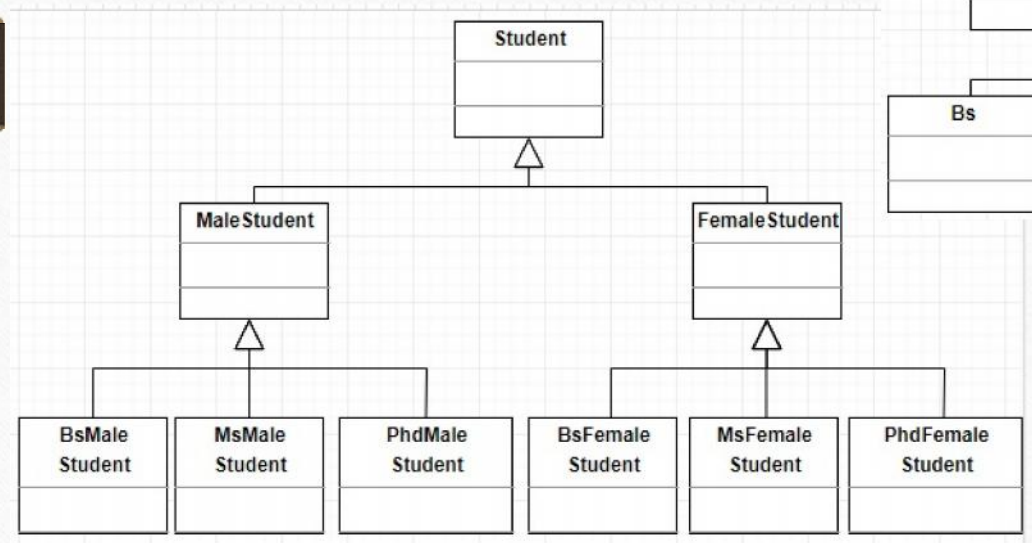
```java
public static void main(String[] args) {
    Student st1 = new Student("Ali Ahmadi", 21, null, 123);
    Student st2 = new Student("Ali Ahmadi", 21, null, 123);
    System.out.println(st1.equals(st2)); // false (before overriding the equals method)
}
```

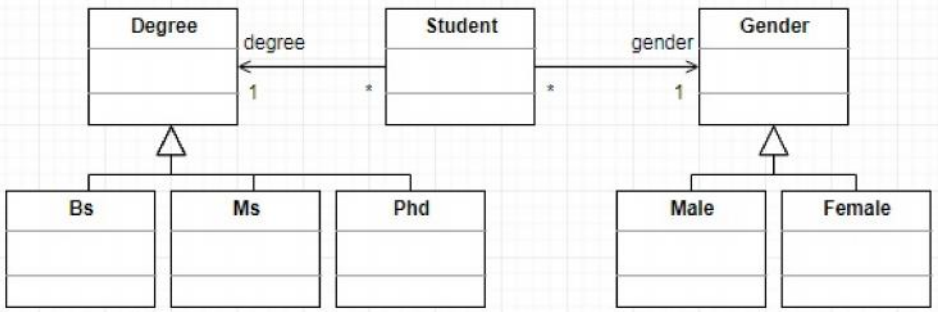
```java
private int studentId;

public Student(String name, int age, Person father, int studentId) {
    super(name, age, father);
    this.studentId = studentId;
}

@Override
public boolean equals(Object obj) {
    if(!(obj instanceof Student))
        return false;
    return studentId == ((Student)obj).studentId;
}
```

نمایش رابطه‌ی وراثت در نمودار کلاس UML



وراثت یا ترکیب؟





تا حد امکان، استفاده‌ی از ترکیب و به شکل کلی رابطه‌ی انجمنی (Association) بر وراثت برتری دارد.