Abstract classes

The INFDEV team

Hogeschool Rotterdam
Rotterdam, Netherlands

Abstract classes

Abstract classes

Abstract classes

The INFDEV team

Abstract classes

Abstract classes

- Interfaces: fully abstract
- Classes: fully concrete
- We want something in the between...

Abstract classes

Abstract classes

The INFDEV team

Abstract classes

About abstract classes

- In OO programming it is possible to design special classes containing some methods with bodies, and some without
- These special classes are called *abstract*

Abstract classes

Abstract classes
The INFDEV team
Abstract classes

About abstract classes

- In the following an abstract class `Weapon` contains a concrete method `GetAmountOfBullets` and an abstract `Fire`
- `Fire` is abstract, since different weapons might come with different kinds of firing

```
1  abstract class Weapon {
2    public int amounOfBullets;
3    public Weapon(int amounOfBullets) {
4      this.amounOfBullets = amounOfBullets;
5    }
6    public int GetAmountOfBullets() {
7      return this.amounOfBullets;
8    }
9    public abstract  void Fire();
10 }
```

Abstract classes

Abstract classes

The INFDEV team

Abstract classes

Which in Java then becomes:

```
1  abstract class Weapon {
2    public int amounOfBullets;
3    public Weapon(int amounOfBullets) {
4      this.amounOfBullets = amounOfBullets;
5    }
6    public int GetAmountOfBullets() {
7      return this.amounOfBullets;
8    }
9    public abstract  void Fire();
10 }
```

Abstract classes

Abstract classes

The INFDEV team

Abstract classes

Instantiating abstract classes

- Not possible directly (what is the result of `new Weapon().Fire()`?)
- Abstract classes have to be inherited in order to use their functionalities
- All abstract methods must eventually get an implementation

Abstract classes

Abstract classes
The INFDEV team
Abstract classes

Implementing our weapon

- In the following a correct implementation of our `Weapon` is provided

```
1  class Gun : Weapon {
2    public Gun(int amounOfBullets) : base(amounOfBullets) {
3    }
4    public override void Fire() {
5      amounOfBullets = (amounOfBullets - 1);
6    }
7  }
8  class FastGun : Weapon {
9    public Gun(int amounOfBullets) : base(amounOfBullets) {
10   }
11   public override void Fire() {
12     amounOfBullets = (amounOfBullets - 1);
13     amounOfBullets = (amounOfBullets - 1);
14   }
15 }
```

Abstract classes

Abstract classes

The INFDEV team

Abstract classes

Which in Java then becomes:

```
1  class Gun extends Weapon {
2    public Gun(int amounOfBullets) {
3      super(amounOfBullets);
4    }
5    public void Fire() {
6      amounOfBullets = (amounOfBullets - 1);
7    }
8  }
9  class FastGun extends Weapon {
10   public Gun(int amounOfBullets) {
11     super(amounOfBullets);
12   }
13   public void Fire() {
14     amounOfBullets = (amounOfBullets - 1);
15     amounOfBullets = (amounOfBullets - 1);
16   }
17 }
```

Abstract classes

Abstract classes

The INFDEV team

Abstract classes

Considerations

- Abstract classes are a mean to combine polymorphism with concrete implementations

This is it!

Abstract classes

The INFDEV team

Abstract classes

The best of luck, and thanks for the attention!