Lessen Java: Reeks 4

David Blinder
Jan G. Cornelis

Inheritance

Via het systeem van **inheritance** (overerving) kan je klassen specialiseren (keyword **extends**)

Als klasse B klasse A extends, neemt het alle attributen methodes over van de moederklasse.

In B kun je attributen en methodes bijmaken, of bestaande methodes overschrijven (override)

```
public class Truck extends Car {…
```

Voorbeeld: Car

```
public class Car {
        final String brand,
        String license_plate;
        double speed, acceleration, fuel;
        int gear;

        Car(String brand, String license_plate) {
                this.brand = brand;
                this.license_plate = license_plate;
        }

        void toggle_engine() {…}
        void set_gear(int x) {…}
}
```

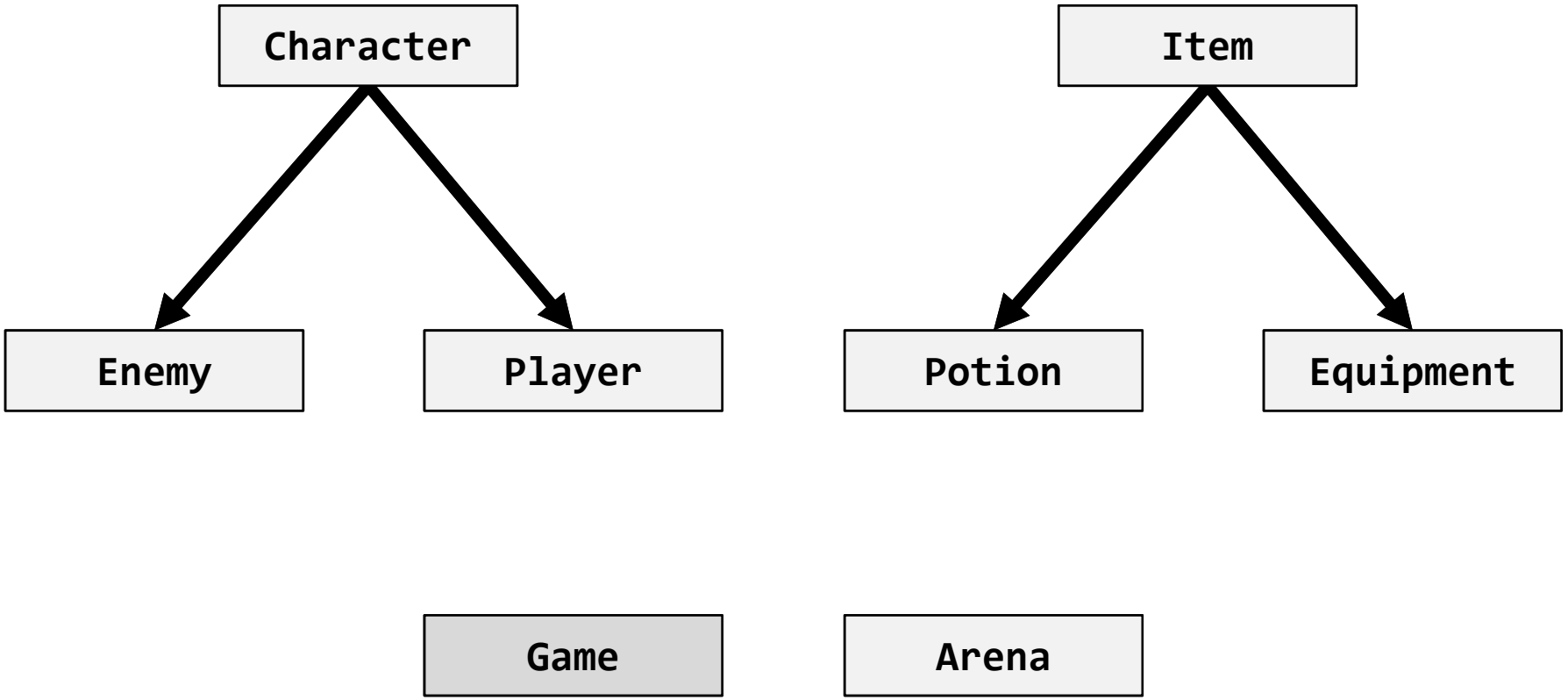
Voorbeeld: Car

```
public class Truck extends Car {
        double carry_capacity;

        Truck(String brand, String license_plate, double
capacity) {
        super(brand, license_plate);
        this.carry_capacity = capacity;
        }

        void drop_container(); // New method
        void set_gear(int x) { // Overridden method
                // ...
                super.set_gear(x);
                // ...
        };
}
```

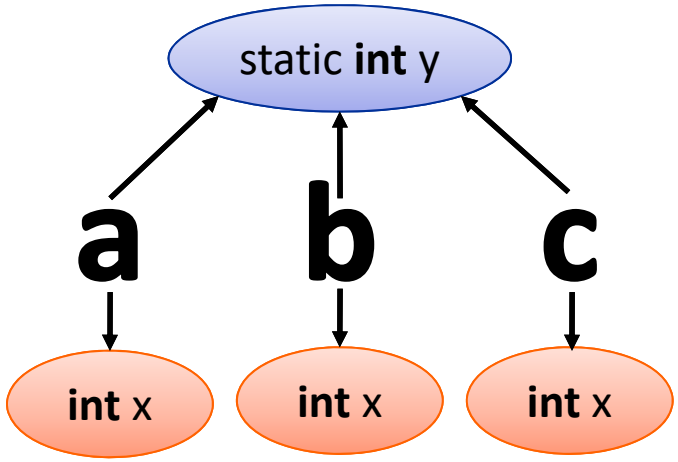
Encapsulatie: public, protected en private

```
public class Car {
        public int x;
                // iedere klasse kan x accessen
        protected int y;
                // alleen Car en zijn subklassen kunnen y accessen
                // + alle klassen binnen dezelfde package
        private int z;
                // je kan z enkel accessen binnen Car zelf
}


public class Truck extends Car {
        …
        y += 5;
}
```

Klassenstructuur



Static




```
public class Num {
        int x = 0;
        static int y = 0;

        public static void main(String[] args) {
                Num a = new Num(), b = new Num(), c = new Num();
                a.x++; b.x++; c.x++;
                a.y++; b.y++; c.y++;

        System.out.println("a.x = " + a.x + " & a.y = " + a.y);
        }
}
```

Output: a.x = 1 & a.y = 3

Inheritance: abstract class

Een abstracte klasse kun je extenden, maar niet instantiëren. Dit komt van pas wanneer je functionaliteit wil delen met **meerdere** dochterklassen, zonder dat de moederklasse op zichzelf mag bestaan.

```
public abstract class Animal {
        String name;

        public void eat() {…}
        public abstract void make_sound();
                // dochterklasse moet make_sound() implementeren
}

Animal A;                       // toegelaten
~~Animal B = new Animal();~~       // mag niet
Animal C = new Dog();           // als (Dog extends Animal) geldt
```

Map

Map<K, V>: een structuur die keys (van klasse **K**) “mapt” op values (van klasse **V**).

Voorbeelden:

- **Inventaris:** Map<Item, Integer>
  (Object → Aantal)
- **Telefoonboek:** Map<String, String>
  (Persoon → Telefoonnummer)

Nuttige methodes:

- V **get**(K)
- void **put**(K, V)
- int **size**()
- Set<K> **keySet**()
- V **replace** (K,V)
- boolean **containsKey**(K)
- V **remove**(K)

Map: voorbeeld

Voorbeeld: een mailinglijst

```
Map<Persoon, String> mailinglijst =  new HashMap<Persoon, String>();

Persoon tim = new Persoon("Tim", "Peeters");
Persoon jana = new Persoon("Jana", "De Slager");

mailinglijst.put(tim, "tim12345@yahoo.com");
mailinglijst.put(jana, "jana.deslager@gmail.com");
mailinglijst.containsKey(tim); // --> returns true
mailinglijst.replace(tim, "tim.peeters@vub.ac.be");
```

Scanner

Een **Scanner** is een klasse die eenvoudige operaties op streams van characters kan toepassen.

Deze kan gebruikt worden om invoer van de gebruiker te verkrijgen.

Voorbeeld:

```
Scanner user_input = new Scanner(System.in);
String input = user_input.next();
int input = user_input.nextInt();
```

Switch statement

Als je vele opeenvolgende if-then-else statements hebt, maak je beter gebruik van **switch**. Dit is leesbaarder, en ondersteunt meerdere executiepaden.

```
switch (var) {
        case 1:
                System.out.println("A");
                break;
        case 2:
                System.out.println("B");
        case 3:
                System.out.println("C");
                break;
        default:
                System.out.println("D");
                break;
        }
```

Continue en Break

```
for (int i=0; i<100; i++) {
        if (i%7 == 0)
                continue;
        System.out.println(i);
}
```

```
String find = "abc"
for (String s : list) {
        if s.equals(find) {
                // ...
                break;
        }
}
```