== vs equals

== vs equals

==
// porównuje referencje w pamięci

equals
// porównuje stan obiektów // wykorzystuje metodę `equals` z obiektu (jeśli obiekt jej nie deklaruje, to z rodzica, jeśli nie ma rodzica to z klasy `Object`)

equals()

Co zostanie wypisane na konsolę?

Zakładając że metoda `equals()` nie jest nadpisana?

```
Card card1 = new Card( name: "Karta",  number: "123");
Card card2 = new Card( name: "Karta",  number: "123");

System.out.println(card1.equals(card2));
```

equals()

Co zostanie wypisane na konsolę?

Zakładając że metoda `equals()` nie jest nadpisana?

```
Card card1 = new Card( name: "Karta",  number: "123");
Card card2 = new Card( name: "Karta",  number: "123");

System.out.println(card1.equals(card2));
```

false

equals()

Dlaczego?

Ze względu na implementację metody `equals()` w klasie Object:

```
public boolean equals(Object obj) {
    return (this == obj);
}
```

Kiedy trzeba nadpisać equals()?

`equals()` należy nadpisać zawsze jeśli klasa reprezentuje dane,

np. tak jak klasa `Card`

W innym przypadku (np. jeśli klasa wykonuje tylko logikę) nie nadpisujemy `equals()`

Zasady implementacji equals()

Metoda `equals()` musi być:

// **zwrotna** → dla każdego `object.equals(object)` zwraca `true`

// **symetryczna** → jeśli
`x.equals(y) == true` to `y.equals(x) == true`

// **przechodnia** → jeśli
`x.equals(y) == true` i `y.equals(z) == true` to
`x.equals(z) == true`

// **spójna** → wielokrotne wywołanie metody na obiekcie daje ten sam wynik (jeśli obiekt nie został zmodyfikowany)

// **object.equals(null) == false**

Przykładowa implementacja equals()

Można wygenerować w IntelliJ IDEA:

`Alt + Insert -> equals() and hashCode()`

```
@Override
public boolean equals(Object o) {
    if (this == o) return true;
    if (o == null || getClass() != o.getClass()) return false;
    CreditCard that = (CreditCard) o;
    return Objects.equals(name, that.name) &&
            Objects.equals(number, that.number) &&
            Objects.equals(balance, that.balance);
}
```

Przykładowa implementacja hashCode()

Zwraca wartość liczbową.

Dobrze jeśli zwraca różne wartości dla obiektów które nie są równe. Używana w kolekcjach wykorzystujących tablice hashujące (https://stackoverflow.com/questions/730620/how-does-a-hash-table-work) np. `HashMap`

Można wygenerować w IntelliJ IDEA:

`Alt + Insert -> equals() and hashCode()`

```
@Override
public int hashCode() {
    return Objects.hash(name, number, balance);
}
```

Zasady implementacji hashCode()

Metoda `hashCode()` musi być:

- **spójna** → wielokrotne wywołanie metody na obiekcie daje ten sam wynik (jeśli obiekt nie został zmodyfikowany)
- **jeśli `x.equals(y) == true`** to **`x.hashCode() == y.hashCode()`**

Z tego powodu jeśli nadpisujemy `equals()` to musimy nadpisać również `hashCode()`

Pytania rekrutacyjne

1. Czy `hashCode()` może zwracać taki sam wynik dla różnych obiektów?

2. Jeśli `hashCode()` zwraca tą samą liczbę dla dwóch obiektów to muszą one być równe?

3. Czy dla obiektów które nie są równe metoda `hashCode()` musi zwrócić inny wynik?

4. Czy dla dwóch równych obiektów metoda `hashCode()` zwraca taki sam wynik?

5. Czy podając jako argument metody `equals()` null poleci wyjątek `NullPointerException`?

Pytania rekrutacyjne

1. Czy `hashCode()` może zwracać taki sam wynik dla różnych obiektów? **TAK**

2. Jeśli `hashCode()` zwraca tą samą liczbę dla dwóch obiektów to muszą one być równe? **NIE**

3. Czy dla obiektów które nie są równe metoda `hashCode()` musi zwrócić inny wynik? **NIE**

4. Czy dla dwóch równych obiektów metoda `hashCode()` zwraca taki sam wynik? **TAK**

5. Czy podając jako argument metody `equals()` null poleci wyjątek `NullPointerException`? **NIE**