Java training

Overriding `toString()`, `equals()` and `hashCode()`

Session overview

- Method overriding definition
- Overriding `toString()`
- Overriding `equals()`
- Overriding `hashCode()`

Method overriding definition, Object class

- **Overriding** - creating a method in a subclass with the same signature as the superclass
  - Signature = name + number and type of parameters + return type
- **Object** class - the superclass for all the classes created in Java
  - Main overriden methods from the Object class:
    - `toString()` → returns a String representation of an object
    - `equals()` → compares the equality of two objects
    - `hashCode()` → returns the object's hashcode (further discussed)

Object's toString()

- `toString()` → returns the String representation of a class
- By default, the `Object`'s class `toString()` method displays:
  - The invoking class name and package
    +
  - @
    +
  - A String representation of the class's hashcode, as an unsigned integer in Base 16
- Example:

  `com.example.domain.Product@2503dbd3`

Overriding `toString()`

- In order to override toString() in a class, we must implement the method:

```
class Product {
    private int id; // + getter and setter
    private String name; // + getter and setter

    public String toString() { // must maintain the same signature
        return id + “, “ + name;
    }
}
```

Hands-on →

Object's equals()

- `equals(Type t)` → tests if the object T is equal to the invoking object
- Without overriding it, `equals()` tests if two objects *have the same reference*
- Example:

```
class Product {
    private String name; // + constructor and getter
}

Product first = new Product("First product");
Product second = new Product("Another product");

System.out.println(first.equals(second)); // returns false
System.out.println(first.equals(first));  // returns true
```

Overriding `equals()`

- Overriding `equals()` - defining the equality for objects from that class:

```
class Product {
    private int id;
    private String name; // + constructor and getters

    public boolean equals(Object other) {
        Product product = (Product) other;
        return (this.id == other.id && name.equals(other.name));
    }
}
```

The equality condition

Hands-on →

Object's hashCode()

- `hashCode()` - method used to compute an object's hashcode (/ signature)
- Main usability - Hash* collections retrieving efficiency (further presented)
- `Object`'s implementation:

  `public native int hashCode();` → `native`: the memory address
- `String` implementation:

$$h(s) = \sum_{i=0}^{n-1} s[i] \cdot 31^{n-1-i}$$

s[0]*31^(n - 1) + s[1]*31^(n - 2) + ... + s[n - 1]

- `hashCode()` recommendation / best practice

Overriding hashCode()

- Implementing hashCode() - defining the used fields & computation method:

```
class Product {
    private int id;
    private String name; // + constructor and getters

    public int hashCode(Object other) {
        int result = id;
        result = 31 * result + name.hashCode();
        return result;
    }
}
```

Hands-on →

Q & A session

1. You ask, I answer
2. I ask, you answer
    a. What is the usefulness of the `toString()` method?
    b. When should we use `equals()`? Do we need to override it?
    c. Why is `hashCode()` useful?