Java training

Types of classes - immutable, wrapper, external, internal, local, anonymous

Session overview

- Immutable classes
- Wrapper classes
- External classes
- Internal classes
- Local classes
- Anonymous classes

Immutable classes

- **Immutable** object - object whose state cannot be changed, after it is created
- Immutable classes - classes designed to be immutable
- Main benefits:
  - Good programming practice, as it leads to simpler code
  - Avoid (accidental / intentional) changes to their values, after they have been set
    - Especially useful for concurrent programming
  - Elimination of code needed to protect mutable objects from corruption
  - Decreased overhead due to garbage collection

Immutable class example

```
public class Tablet {
    private final String name;

    public Tablet(String name) {
        this.name = name;
    }

    public String getName() {
        return this.name;
    }
}
```

The `final` modifier will make the field setting mandatory

Constructor with parameters (sets all the class fields)

Only the getter is needed, as there is no need to further set the value

Using the immutable class

```
Tablet tablet = new Tablet(“Useful”);
```
→ must use that constructor

```
System.out.println(tablet.getName);
```
→ can only use the setter

If any internal value needs to be changed → a new object must be created

- Main drawback towards the usage of immutable classes (for some people)

Wrapper classes

- There are 8 primitive types in Java:
  - `boolean`
  - `byte`
  - `char`
  - `int`
  - `float`
  - `double`
  - `long`
  - `short`
- Each primitive type has a wrapper class for it →

Wrapper classes summary

Primitive type	Wrapper class	Constructor argument
`boolean`	`Boolean`	`boolean` or `String`
`byte`	`Byte`	`byte` or `String`
`char`	`Character`	`char`
`int`	`Integer`	`int` or `String`
`float`	`Float`	`float`, `double` or `String`
`double`	`Double`	`double` or `String`
`long`	`Long`	`long` or `String`
`short`	`Short`	`short` or `String`

External classes

- External classes - classes not bundled in the JDK (Java Development Kit)
- All projects use external classes, as they offer already developed features:
  - Web programming support
  - Logging frameworks
  - Database access
  - Interactions with other systems
  - Many other use-cases
- Will be further discussed when we'll discuss the build tools

Internal classes

- Internal classes - classes which have package-private (default) access
- Example:

```
package com.sample;

class InternalClassExample {
    // fields and methods
}
```

Local classes

- Classes defined in a block, typically in the body of a method
  - Other possible places: a `for` or `while` loop, an `if` clause
- Example:

```
class ProductValidator {
    public void validate(String product) {
        class Product {
            int id, String name; → + code to parse the String into a Product
        }
    }
}
```

Anonymous classes

- Classes declared and available inside a block, usually:
  - A method
  - A `for` or a `while` loop, an `if` clause
- Main difference between local classes - they don’t have a name
- They are useful to make the code more concise

Example →

Anonymous class example

```
class ProductValidator {
    interface Validator {
        boolean validateProperty(String value);
    }
    public void validate(String product) {
        new Validator() {
            boolean validateProperty(String value) {
                // code to validate the specified value
                return true; // is valid
            }
        }
    }
}
```

Interfaces - presented soon

Anonymous class example

Q & A session

1. You ask, I answer
2. I ask, you answer