Encapsulation + Constructors

Josh Wilcox (jw14g24@soton.ac.uk)

February 3, 2025

Contents

1 Encapsulation in Java

- The **bundling of data** - with methods that operate the data
- Prevents direct access from external classes
- Makes variables only accessible *within* the specific class

1.1 Definition

```
private Data_Type Var_Name;
```

- Once a data value is `private`, other classes will not be able to access it

1.2 Setters and Getters

- Special functions within a class that edit or fetches the Private variables within each class
- Affords the guarantee that only valid data is set into the properties
- Could also control how the value is actually fetched

2 Benefits of Encapsulation

- Supports the idea that a class should be **responsible for managing itself**
- Each class should present a **clean interface**
  - Hides messy or unneeded methods or data that is only needed internally
  - Prevents inner workings of the class from being externally revealed
- Results in cleaner, more portable code that is more maintanable

3 Constructors

```
class class_name {
    public class_name(parameters){
        // Statements
        // No Returns
    }
}
```

- Has to be named exactly the same as the class name (including capital)
- No Return Type - **Do not Return anything**
- No Keyword return
- The code initialises the objects and set critical member variables
- *maintains encapsulation*
- Can be overloaded

3.1 Default Constructor

- When there is no constructor, java creates a **default constructor** instead.
- Takes no parameters and contains no behaviour, just creates default Object instances

```java
public class Example {
    int number;          // int primitive
    boolean flag;        // boolean primitive
    String text;         // Reference type (String)

    // No constructor defined, so the compiler provides a default constructor
}

public class Main {
    public static void main(String[] args) {
        Example obj = new Example();
        System.out.println(obj.number);  // Outputs: 0
        System.out.println(obj.flag);    // Outputs: false
        System.out.println(obj.text);    // Outputs: null
    }
}
```