Constructors

zyBook Chap 9.4, 9.5, 9.9

Constructor

Constructor → A special method that **initializes the state** of new objects as they are created:

- Constructor's name **matches** the **class name**
- **NO return type**
- **Parameters** are used to **specify the object's initial state**
- Access object's fields and methods directly

Constructor – Book

- **Initializing the fields** in the constructor in the Book Class

```java
public class Book {

    // Instance variables (Fields)
    private String title;
    private String author;
    private int pubYear;
    private String checkedOutBy;
    private int location;
    private Date dueDate;

    // Constructor
    public Book(String title, String author,
                int pubYear, int location) {

        this.title = title;
        this.author = author;
        this.pubYear = pubYear;
        this.location = location;
        this.checkedOutBy = null;
        this.dueDate = null;
    }
}
```

Constructor – Book

- Initializing the fields in the constructor in the Book Class

```
public Book(String title, String author, int pubYear, int location) {
    this.title = title;
    this.author = author;
    this.pubYear = pubYear;
    this.location = location;
    this.checkedOutBy = null;
    this.dueDate = null;
}
```

- Constructing a Book object with the **specified constructor** in the **client program**

```
Book book = new Book("Building Java Programs, 5th Edition",
                     "Stuart Reges, Marty Stepp", 2020, 7);
```

Method Overloading

- Method overloading is a feature that allows a class to have **more than one method** with the **same name**, but with **different sets of parameters**

Constructor Overloading

- A class can have **multiple constructors** and each one must accept a **unique set** of **parameters**
  - Paradigm → **One constructor contains true initialization code** – all other constructors call it with the keyword `this`

```
public Book(String title, String author, int pubYear, int location) {
    this.title = title;
    this.author = author;
    this.pubYear = pubYear;
    this.location = location;
    this.checkedOutBy = null;
    this.dueDate = null;
}

public Book(String title, String author, int pubYear) {
    this(title, author, pubYear, 0); // calling the first constructor
}
```

this is used used to call one constructor from another

Common Constructor Bugs

Re-declaring fields as local variables ("shadowing")

- This code declares local variables with the same name as the fields, rather than storing values into the fields. **The fields remain default value**

```
import java.util.Date;

public class Book {
    private String title;
    private String author;
    private int pubYear;
    private String checkedOutBy;
    private int location;
    private Date dueDate;

    public Book(String newTitle, String newAuthor,
                int newPubYear, int newLocation) {
        String title = newTitle;
        String author = newAuthor;
        int pubYear = newPubYear;
        int location = newLocation;
        String checkedOutBy = null;
        Date dueDate = null;
    }
}
```

Common Constructor Bugs

Giving the constructor a **return type**

- This is not a constructor, but an **instance method** named `Book`.

```
import java.util.Date;

public class Book {
    private String title;
    private String author;
    private int pubYear;
    private String checkedOutBy;
    private int location;
    private Date dueDate;

    public void Book(String title, String author,
                     int pubYear, int location) {
        this.title = title;
        this.author = author;
        this.pubYear = pubYear;
        this.location = location;
        this.checkedOutBy = null;
        this.dueDate = null;
    }
}
```

Default Constructor

- If no constructors are specified in the class, the **default constructor** exists that **takes no parameters**. It will initialize fields to their **default values**.
  - `<ClassName> a = new <ClassName>();`

```
// Default constructor that takes no parameters
Book book = new Book();

// Initialize the fields with the setter methods
book.setTitle("Building Java Programs, 5th Edition");
book.setAuthor("Stuart Reges, Marty Stepp");
book.setPubYear(2020);
book.setLocation(7);
```

Constructor with No Parameters

Once we write a constructor, we can no longer use the default constructor (constructor automatically supplied by Java with no parameters)

- If we need a constructor with no parameters in addition to the other constructors we create, we need to define it ourselves.
  - Only create a constructor with no parameters if it makes sense for your object.