Setters

Yegor Bugayenko

Lecture #1 out of 8
90 minutes

- Mutability
- Problems
- Read and Watch

Chapter #1:

Mutability

Which object is immutable?

```
1  class Book {
2    private String title;
3    Book(String t) { title = t; }
4    void setTitle(String t) {
5      this.title = t;
6    }
7    String getTitle() {
8      return this.title;
9    }
10 }
11 Book b = new Book("Object Thinking");
12 b.setTitle("It");
```

```
1  class Book {
2    private final String title;
3    Book(String t) { title = t; }
4    void withTitle(String t) {
5      return new Book(t);
6    }
7    String getTitle() {
8      return this.title;
9    }
10 }
11 Book b1 = new Book("Object Thinking");
12 Book b2 = b1.withTitle("It");
```

There are four gradients of immutability

I. Constant

II. Not a Constant

III. Represented Mutability

VI. Encapsulated Mutability

Gradient I: Constant

```
class Book {
  private final String t;
  Book(String t) { this.t = t; }
  String title() {
    return this.t;
  }
}
```

```
Book b = new Book("Object Thinking");
String t1 = b.title();
String t2 = b.title()
```

Gradient II: Not a Constant

```
class Book {
  private final String t;
  Book(String t) { this.t = t; }
  String title() {
    return String.format(
      "%s / %s", title, new Date()
    );
  }
}
```

```
Book b = new Book("Object Thinking");
String t1 = b.title();
String t2 = b.title()
```

Gradient III: Represented Mutability

```
class Book {
  private final Path path;
  Book(Path p) { this.path = p; }
  Book rename(String title) {
    Files.write(
      this.path,
      title.getBytes(),
      StandardOpenOption.CREATE
    );
    return this;
  }
  String title() {
    return new String(
      Files.readAllBytes(this.path)
    );
  }
}
```

```
Book b = new Book("Object Thinking");
String t1 = b.title();
b.rename("Elegant Objects");
String t2 = b.title()
```

Gradient VI: Encapsulated Mutability

```
class Book {
  private final StringBuffer buffer;
  Book rename(String t) {
    this.buffer.setLength(0);
    this.buffer.append(t);
    return this;
  }
  String title() {
    return this.buffer.toString();
  }
}
```

```
Book b = new Book("Object Thinking");
String t1 = b.title();
b.rename("Elegant Objects");
String t2 = b.title()
```

Chapter #2:

Problems

Side effects

With a side effect:

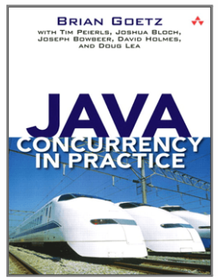
```
1  public String post(Request request) {
2    request.setMethod("POST");
3    return request.fetch();
4  }
5
6  Request r = new Request("http://...");
7  r.setMethod("GET");
8  String first = this.post(r);
9  String second = r.fetch();
```

Without a side effect:

```
1  public String post(Request request) {
2    return request
3      .withMethod("POST")
4      .fetch();
5  }
6
7  Request r = new Request("http://...")
8    .withMethod("GET");
9  String first = this.post(r);
10 String second = r.fetch();
```

Thread (un-)safety

```
class Books {
  private int c = 0;
  void add() {
    this.c = this.s + 1;
  }
}
```



Goetz et al. explained the advantages of immutable objects in more details in their very famous book “Java Concurrency in Practice” (highly recommended!)

```
ExecutorService e =
  Executors.newCachedThreadPool();
final Books books = new Books();
for (int i = 0; i < 1000; i++) {
  e.execute(
    new Thread(
      () -> {
        books.add();
      }
    )
  );
}
// What is the value of "books.c"?
```

Object Relational Mapping (ORM)

```
class BookDTO {
  private int id;
  private String author;
  private String title;
  BookDTO(int i, String a, String t)
    { id = i; author = a; title = t; }
  int getId() { return id; }
  String getAuthor() { return author; }
  String getTitle() { return title; }
}
```

```
class Books {
  BookDTO findById(int id) { /* ... */ }
}
BookDTO dto = books.findById(42);
dto.getTitle();
dto.getAuthor();
```

Chapter #3:

Read and Watch

Objects Should Be Immutable by me

Gradients of Immutability by me

Immutable Objects Are Not Dumb by me

How an Immutable Object Can Have State and Behavior? by me

How Immutability Helps by me