Chapter 4: Reference Data Types

Section 4.1: Dereferencing

Dereferencing happens with the `.` operator:

```
Object obj = new Object();
String text = obj.toString(); // 'obj' is dereferenced.
```

Dereferencing *follows* the memory address stored in a reference, to the place in memory where the actual object resides. When an object has been found, the requested method is called (`toString` in this case).

When a reference has the value `null`, dereferencing results in a NullPointerException:

```
Object obj = null;
obj.toString(); // Throws a NullpointerException when this statement is executed.
```

`null` indicates the absence of a value, i.e. *following* the memory address leads nowhere. So there is no object on which the requested method can be called.

Section 4.2: Instantiating a reference type

```
Object obj = new Object(); // Note the 'new' keyword
```

Where:

- `Object` is a reference type.
- `obj` is the variable in which to store the new reference.
- `Object()` is the call to a constructor of `Object`.

What happens:

- Space in memory is allocated for the object.
- The constructor `Object()` is called to initialize that memory space.
- The memory address is stored in `obj`, so that it *references* the newly created object.

This is different from primitives:

```
int i = 10;
```

Where the actual value `10` is stored in `i`.