Introduction to Objects

James Brucker

Software Design and OOP

Object-oriented design and programming is the dominant paradigm in software development.

To understand *why* you first need to understand objects and classes.

What is an Object?

An object is a program element that *encapsulates* both data and behavior.

An object contains both data and the methods that operate on the data.

An object can control what information it exposes to the outside, and what it hides.

Example

String object: `s = new String("i am an object")`

- data: the characters in the String
- methods: toLowerCase, substring, indexOf, ...

Scanner object: `console = new Scanner(System.in)`

- data: in *input source* the scanner is reading, the current position in the input, the separator
- methods: hasNext, next, hasNextInt, nextInt, ...

Conceptual meaning of Objects

Objects represent "things" in the problem domain.

Examples:

Banking app:	**money**
	bank account
	customer
Board game:	**board**
(chess)	**game piece**
	player

Objects - give examples

What are *kinds of things* you would find in a **Restaurant Application**?

____

____

____

____

____

____

Objects and Classes

A **class** defines a **kind of object**.

The class defines:

1. attributes - to hold the data an object knows
2. methods - object's behavior (what it can do)
3. constructors - how to initialize a "new" object

String Class and Object

Consider a String object: `String s = "Hello";`

String <u>class</u> defines

attributes - what the String ***knows*** (also called ***fields***)

methods - what the String can ***do*** (its ***behavior***)

String <u>object</u> (s) defines the values of attributes (data)

<u>s: String</u>

length = `5`

value= `['H','e','l','l','o']`

---

length( )

charAt(int)

substring(start, end)

toLowerCase( )

attributes are information an object remembers or stores
Also called: fields

methods are what the object can do.
Also called *behavior*

`new` - Creates object from a class

"new" creates a new object.

"new" invokes a *constructor* to initialize the object's attributes.

Example: create some Date objects

```
// constructor with no parameters - current date
Date now =  new Date( );
// constructor with 3 parameters - specify a date
Date ny = new Date(105,Calendar.JANUARY,1);
System.out.println( now ); // 24 Oct 2017, 14:05:32
System.out.println( ny );  // 01 Jan 2005, 00:00:00
```

Each object has its own attributes

Each object has its own copy of the attributes.

Changes to one object do no modify attributes of other objects.

```
Date now = new Date( ); // today is 24 Oct 2017
Date now2 = new Date( );
now2.setMonth( Calendar.DECEMEBER );
now2.setDate( 1 );
now.setHour( 12 );
System.out.println( now2 ); // 01 Dec 2017, 12:32
System.out.println( now );  // 24 Oct 2017, 14:32
```

Class can have many Constructors

Scanner class has many constructors. See the Javadoc.

```
// Scanner for reading InputStream
Scanner s1 = new Scanner( System.in );
// Scanner for parsing a String
Scanner s2 = new Scanner("Parse me, man.");
// Scanner opens and reads a File object
File file = new File("/etc/passwd");
Scanner s3 = new Scanner( file );
```

Default Constructor

A constructor that has no parameters.

Also called "no argument constructor".

Not all classes have a default constructor!

```
// An empty ArrayList object (default constructor)
ArrayList list1 = new ArrayList();
// ArrayList object with data copied from array
String[] arr = "To data or not to data?".split(" ");
ArrayList<String> list2 = new ArrayList<>( arr );
// Error: Scanner does not have default constructor
Scanner scanner = new Scanner( );  // ERROR
```

State

An object has "**state**", which is defined by the value of its attributes.

State may also be defined by things an object is related (or connected) to, such as a file or InputStream.

Examples of State:

LightBulb object - state is "off" or "on"

FileInputStream object - open or closed, data in a file

Scanner object - the delimiter pattern (default is space) and its position in the input source.

3 Characteristics of Objects

Objects have:

Behavior - what an object can do. Defined by methods.

State (or **Knowledge** or **Data)** - what an object knows. Defined by attributes

Identity - objects are unique, even if they have the same type and attribute values.

Please Memorize These

Invoking Behavior (Methods)

To invoke a method of an object, write:

```
variable.method( )
```

A variable that *refers* to the object

A method that belongs to the object

```
> String s = "Hello Dog";
> s.length()
9
> s.toUpperCase()
"HELLO DOG"
> s.substring(0,5)
"Hello"
```

Class defines a kind of object

Memorize this.

Definition:

"A **class** is a **blueprint** or **definition** for a ***kind*** of object**."**

`Sale` class defines:

- attributes of a sale.
- behavior (methods) of a sale.
- how to create and initialize a sale.

Objects are *distinct*, even if same value

Identity: Objects are distinct

Each time you call "new" it creates a new object.

```
String s1 = new String("OOP");
String s2 = new String("OOP"); //same data

// are they same object?
```

$$\texttt{System.out.println}\underbrace{\texttt{(s1 == s2)}}_{\textbf{FALSE}}\texttt{;}$$

Review

1. What is the definition of a **class** in OOP?

2. What are the **3 characteristics of objects**?

3. How do you create a Date object for the date Feb 15, 2000?

4. Is this true or false? Why?

```
Double x = new Double(1.0);
Double y = new Double(1.0);
(x == y)
```

Next

1. Exercise: create Scanner objects that read the same file.

   Purpose: to see that each object has its own state

2. How to define your own classes.

3. Other ways to create objects - "new" is for newbies