Summary 2 – Constructing and Using Objects

Constructing objects

Instances of a class are called ***objects*** and are *constructed* by using the *new* operator, e.g.

```
Rectangle box = new Rectangle(5, 10, 20, 30);
```

Several steps are happening here:

1. Java reserves space for a Rectangle object.
   - As much space as specified by the fields of the Rectangle class
2. Rectangle's *constructor* runs.
   - It uses the data passed to it (here, the left/top corner of the rectangle, its width and its height) to initialize the fields of the Rectangle object.
   - It does any other initialization required for constructing a Rectangle.
3. Java reserves space for a variable *box* that can refer to Rectangle objects.
4. *box* is set to refer to the object that was constructed.

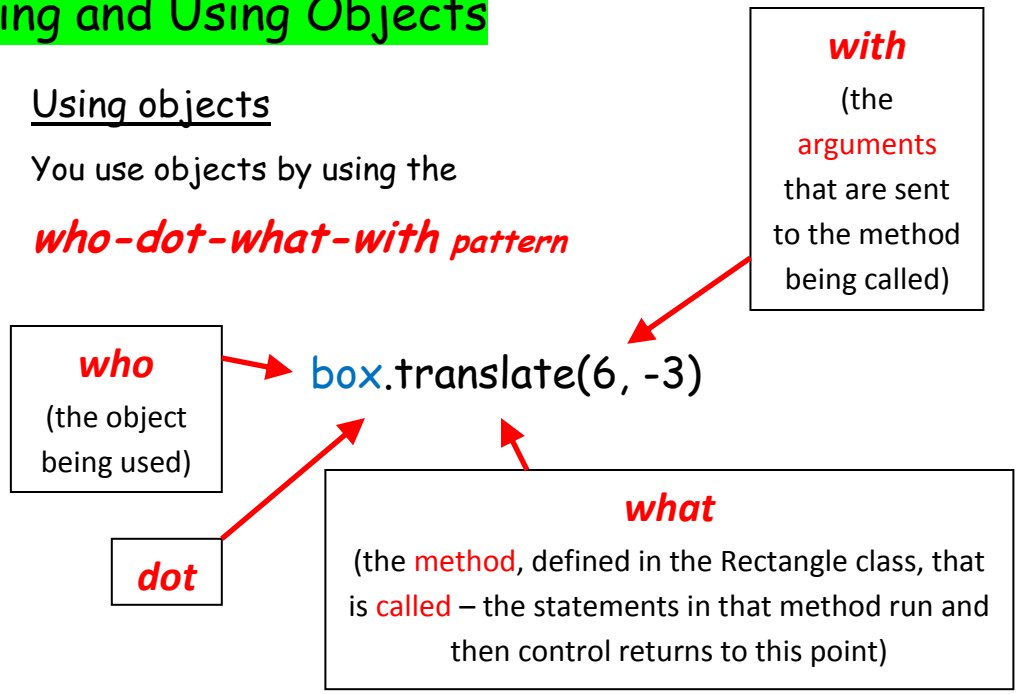
Note:

- The name of the constructor is always the same as the name of the class.
- You can have constructors with different signatures (i.e., different types/number of parameters). For example, Rectangle has a constructor that takes nothing and constructs an empty rectangle at (0, 0).

Using objects

You use objects by using the

who-dot-what-with pattern



Notes:

- The *what* can be a field (data associated with the object). A method call always has parentheses (even if there are no arguments); that's how you tell it's a method call.
- In Eclipse, after you type the dot, Eclipse shows you all the methods and fields available to you!
- If *who* is a class name (instead of an instance of a class), then the *what* method/field must be *static* – see the Summary on *static*.

- Example (from JavaEyes)

```
JavaEyesFrame frame = new JavaEyesFrame();
frame.setVisible(true);
```

- For further study:
  - *Big Java*, chapter 2 *Using Objects*
  - This summary's *author:* David Mutchler
  - See also the Summaries on:
    - *Variables, Types, Classes and Objects*