Lesson 5.03 — Object Initialization: Constructors

Overview

Objectives — *Students will be able to...*

- **Describe** and create classes, objects, and client code.
- **Predict** the output of the code that uses objects.

Assessments — *Students will...*

- **Complete** Practice questions

Homework — *Students will...*

- **Read** BJP 8.4
  - **Take notes**, since you will have to teach a mini-lesson later in this unit

Materials & Prep

- **Projector and computer** (if you are able to/opt to use Eclipse with your students)
- **Whiteboard and markers**
- **Classroom copies** of WS 5.3.1, WS 5.3.2
- **Student small-group assignments** (3-4 per group)
- **2 dialogue bubbles** (index cards or sticky notes) for each group
- **Roll of tape/glue stick** for each group
- **1 large, 1 medium, and 2 small sticky notes** for each group
- **1 blank sheet of paper** for each group

Most teachers will either already have these materials on hand, or be able to borrow them from another teacher or the main office. You should try to give your classroom teacher at least 1 week notice to get these supplies together.

Template 5.3.1 should be re-sized to ledger or legal sized paper, if it is available in your school. If you have access to these larger pieces of paper, encourage students to write their code largely so students can easily read the examples when they are posted around the room.

Pacing Guide

Section	Total Time
Bell-work and attendance	5min
Introduction	10min
Student work	30min
Student viewing & exit ticket	10min

Procedure

This lesson includes two hooks: First, assemble the packets of materials for each group before class, and lay the supplies out for students to see/wonder about. Secondly, introduce today's lesson with a hipster flair (if you can pull it off) by emphasizing the artisanal, custom-made, hand-crafted, boutique (etc.) nature of the classes and objects they will be creating today. The sky is the limit! They can choose any class of objects they are interested in for their group work.

If you feel additional motivation is needed, you can offer a prize (TEALS swag, etc.) to the most creative, complete, and correct code sample. Offer some just-in-time instruction, then let the students work together with the guidance of WS 5.3.1

Bell-work and Attendance [5 minutes]

Introduction [10 minutes]

1. We know that objects use constructors—you might remember them from earlier in the year when we used the new keyword new to construct a new array.
   - Start your constructor with the keyword public
   - Follow with the class name and whatever parameters you think you should include
     - This is a design question—which parameters do you think should be auto-initialized? When does it make sense to proscribe an initial state?
2. Work through the examples we've reviewed in previous classes. What constructor look like for the Student class? The Dog class? A Forecast class?
   - Give the first example, but with each example have students offer increasing amounts of the code themselves.
   - Ask students if they can think of some situations where they might want to leave out some of the fields from the constructor.

```
public class Student {
    public String name;
    public int    gradeLevel;
    public double gpa;

    public Student (String n, int gl, double g) {    // It's a good idea to use a single
        name = n;                                    // letter from the fields you are
        gradeLevel = gl;                             // initializing in the constructor.
        gpa = g;                                     // It keeps things simple!
    }
```

```
public class Dog {
    public String breed;
    public double weightInKg;

    public Dog (String b, double w) {    // Students may ask why we have to create
        breed = b;                       // these additional parameter names. This
        weightInKg = w;                  // plumbing may feel clumsy, but it's just
    }                                    // a step we have to accept in Java.
```

```
public class Forecast {
    public double  windSpeed;
    public String  windDirection;
    public boolean tornadoWarning;

    public Forecast (double ws, String wd) {    // Maybe it doesn't make sense to have
        windSpeed = ws;                         // a tornado warning in the forecast.
        windDirection = wd;                     // This is a design choice!
    }
```

Because you have built a custom class with objects that you designed, you can't rely on Java to auto-initialize your objects to zero values like it does for the Array class. The array class has its own constructor that says "set all initial values to zero-equivalents." You will always need to write your own constructor to initialize your new objects.

Student Work [30 minutes]

Emphasize with students...

Big Ideas - Tools and technologies can be adapted for specific purposes In this activity you and your classmates will be creating your own classes and objects. This is a great example of a how a technology like Object Oriented Programming can be adapted to serve a wide variety of purposes.

As you learn about the other classes and objects created by your classmates, and as you compare it to your own, think carefully about how different people implement technology in different ways. Your classmates will probably present ideas that you had never thought of. That's exciting!

---

1. If you have a sample (that you did yourself, or that you saved from previous students' work), hold it up as an example for the class, but do not let students look too closely. The idea here is for students to see that there are different sheets of paper and steps to the project; you don't want them looking at details of the code.
2. Before you break students into groups, remind the class that they should read through all of the instructions first so they get a good idea of what their "boutique" class is required to do/contain.
    a. They should spend at least 5 minutes discussing their approach to program design, potential strengths, and weaknesses of that design, then come to consensus.
    b. Warn students that you will ask all group members to justify their design choices. (Follow through with this, walking around the room to spot-check students!)
3. Ask students what they should do if they have a question or get hung up on part of the exercise. (Check their notes, check the book, discuss it as a group, and if all that doesn't work, raise their hands for help.)
4. Break students into their assigned teams and distribute the materials (worksheets 5.3.1 and 5.3.2) to each group. Invite them to start on reading and design debate.
5. If students are struggling with a question, you can refer them to the following sections in their book:
    a. Steps 1–2 — The first half of section 8.2
    b. Steps 4–11 — The second half of section 8.2
    c. Step 3 — The first half of section 8.3
    d. Step 12 — Section 8.1

Student Viewing & Exit Ticket [10 minutes]

1. Have students put out their code on desks, or pinned to the wall, and have the entire class visit each Artisanal Class/Object set.
2. As a ticket to leave, have students write down their name, the name of another group's class, and how they would declare an object according to that groups' constructor.

Accommodation and Differentiation

For ELL classes, you may want to let students investigate the sample finished product more closely to give them cues on instructions so they can focus on the coding instead. If the task is still slowing the class down too greatly, read the directions aloud to the class, and demonstrate the step required (e.g. selecting the sticky note and sticking it to the template).

More advanced classes will whiz through this activity in 15 minutes or less. If this happens in your classroom, encourage students to:

- add more methods, including comments, and labeling implicit parameters
- add more client code, including comments on the code to explain what it does
- add other classes and objects (on additional templates) and create a model that shows how all the objects, classes, and client code might link together in a larger program
- reserve some time to complete today's homework and move on to Lesson 5.4.

Teacher Prior CS Knowledge

- Constructor syntax. Here is the statement for creating an object from class Phone:

  ```
  Phone myPhone = new Phone();
  ```

  Here's the constructor for class Phone:

  ```
  public class Phone {
      public Phone () {
          // Phone constructor code goes here
      }
  }
  ```

  If you break down the creation of a Phone object: `new Phone ()` the right hand side `Phone ()` is simply a method call. The method that is called is the constructor. If we now look at the constructor definition `public Phone ()`, it matches the right hand side of the creation of a Phone object. The only difference between calling a constructor when creating an object and a static or non-static method is the return type is implied. A constructor by definition returns a reference to an object in this example, a reference to a Phone. Since the constructor by definition can only return one type, it was removed from the syntax when defining a constructor.

Misconceptions

Students adding void return type to constructor definition. Students are in the habit of specifying a return type for methods. The one case where the return type is not needed is when defining the class' constructor. If a return type is specified, the method is treated as a method of the class and not as a constructor. The code will compile but the constructor will never be called.

Videos

- BJP 8-3, *Defining a Class* http://media.pearsoncmg.com/aw/aw_reges_bjp_2/videoPlayer.php?id=c8-3
- CSE 142, *Using vs building objects* (1:41–7:36) https://www.youtube.com/watch?v=V3Gs1Ug82_E&start=101
- CSE 142, *toString()* (optional) (7:37–23:33) https://www.youtube.com/watch?v=V3Gs1Ug82_E&start=457
- CSE 142, *Constructors* (23:34–30:13) https://www.youtube.com/watch?v=V3Gs1Ug82_E&start=1414
- CSE 142, *Multiple Constructors* (optional) (30:14–35:37) https://www.youtube.com/watch?v=V3Gs1Ug82_E&start=1814

Forum discussion

Lesson 5.03 Object Initialization: Constructors (TEALS Discourse account required)