Lesson 5.02 — Object State & Behavior

Overview

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

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

Assessments — *Students will...*

- **Complete** WS 5.2 individually or in pairs.

Homework — *Students will...*

- **Read** BJP 8.3 up to "The Keyword this"
- **Complete** self-check questions #9-11, 13-16

Materials & Prep

- **Projector and computer**
- **Whiteboard and markers**
- **Classroom copies** of WS 5.2

You should read the introduction on the Bulbepedia website so you understand the main ideas behind the Pokémon game. If you search YouTube, you can find recorded games to see how a Pokémon battle starts, progresses, and ends.

Pacing Guide

Section	Total Time
Bell-work and attendance	5min
Introduction	15-30min
Student practice: WS 5.2	25min
Students trade work, check, and turn in	10min

Procedure

Yesterday you asked students to do some research on what fields, constructors, and methods would be appropriate for building a custom-made Pokémon class. Solicit students' input before you work through a Pokémon class example.

Ask students what they think a Pokémon class should include, and why. Encourage students to argue for or against certain design features in the Pokémon class. Should the class include all Pokémon stats? Are there any behaviors (methods) you think all instance objects should have? What are some examples of instances of the Pokémon class? (Any individual Pokémon is an instance of the Pokémon class, for example Pikachu, Bulbasaur.)

Bell-work and Attendance [5 minutes]

Introduction [15-30 minutes]

1. Ask students to review their notes from the day before, reminding you what the main components of a class are. (Fields, methods, constructors, and encapsulation.)
    - The syntax for declaring a field is the same as the syntax for declaring normal variables (type followed by semicolon). If your students are feeling confident, invite a volunteer up to declare a field on the whiteboard for your Pokémon class.
    - Remind students that fields signify that EVERY instance object of the class should have that variable inside it, so as their example, they should declare a trait that every instance of Pokémon will have.

- Your example should look something like this:

```
public class Pokemon {
    private int hp;       // Pokémon stats include hit points, or "HP"
    private int attack;
```

- Students will probably start volunteering additional examples once they realize that stats make for good fields. Some other fields include:

```
    private int defense;
    private int specialAttack;
    private int specialDefense;
    private int speed;
```

- For the sake of simplicity, try to keep students to 2 or 3 fields for now. Don't just arbitrarily declare this; encourage students to think about how we use classes and objects as models. Ask them to criticize your current model.
  - Does it need to be complex yet?
  - If we opt for simplicity, what are we yielding in sufficiency/robustness?
  - If students agree to keep it simple for now, remind them that they can make a design choice to increase complexity later. Most programmers start with a simpler model and build up as they flesh out their program.

  If students need additional examples for appropriate fields, lead students through the following examples (having them add as much of the code as possible). Make sure that students can justify their choices in fields and explain why they would include some data and not others. At every opportunity, repeat the fact that they are **using data to model the real world**:

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

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

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

2. Remind students that in the previous class they learned that objects combine both state (data) and behavior (methods). So far we've created fields in our classes that state what data will be stored in all instance objects.
   - What would be a good method to include in all instances of the Student class?
   - What would be a good method for all instances of the Dog class to have?
   - What method should all forecasts have, no matter what area you're forecasting for?

3. Let's add a method inside the object that will report information about the data stored in our Pokémon objects. Because this method is being written within the object, we refer to it as an instance method (it is not in client code).
   - Pokémon get an effort ribbon if their combined stats exceed a certain value. What would the method sumStats look like?

```
public int sumStats() {
    return hp + attack + defense + specialAttack + specialDefense + speed;
}
```

- Since this method gets information about your Pokémon instance, but doesn't change any of the values, what do you call this type of method? (*Accessor*)
- Is this client code? (*No, it is part of the Pokémon class, which is why we call it an instance method.*)

4. Let's write another instance method that will let us change the state (data values) stored in our Pokémon instance objects. In the game, what can you do to cause your stats to change? (Win battles, consume vitamins)
   - Pokémon can use vitamins to boost their stats. Here are some example vitamins for you to use at the board:
     - **hpUp:** + points to HP
     - **protein:** + points to attack
     - **iron:** + points to defense
     - **zinc:** + points to specialDefense
   - In keeping with our earlier example, a method to update stats with vitamins would look something like this:

     ```
     public void consumeVitamin (int hpUp, int protein) {
         hp += hpUp;
         attack += protein;
     }
     ```

     Students may want to add other vitamins.

5. Ask students if they can deduce the syntax rules for instance methods based on the two methods we've written so far:

   ```
   public <type> <name> (<type <name>, <type> <name> …) {
       <statement>
       <statement>
       …
   }
   ```

6. Since we know that all instances of our Pokémon class will have initial values to their stats, we could create a constructor to initialize all of our values.
   - It often doesn't make sense to have Java auto initialize our stats to 0, so we build our own constructor that requires us to pass initial parameters.
   - Have students point out to you the class, fields, and constructor:

     ```
     public class Pokemon {
         private int hp;
         private int attack;

         public Pokemon (int hitpoints, int a) {          // In a complete version you
             hp = hitpoints;                              // would include all stats.
             attack = a;
         }
     }
     ```

7. Now that you used the constructor, it's very easy to create objects! What would an instance of the class Pokémon be? (Any Pokémon type; Pikachu, Bulbasaur, Squirtle, etc.)

   ```
   Pokemon pikachu = new Pokemon(70, 120);
   ```

   If students are getting excited about this example, ask them to look up types and their typical initial values (IVs) for hit points and attack. Let them practice constructing new instances of the Pokémon class.
   - Ask students how you would add the Pokémon type (electric, ground, rock, etc.) to the constructor.

- Point out that it is incorrect to construct a Pokémon object without passing initial hitPoints and attack parameters. Since you wrote a custom-made constructor for your class, Java won't let you call new Pokémon() anymore. Instead, your code just won't compile.

Student Practice: WS 5.2 [25 minutes]

Emphasize with students...

Big Ideas - Personal design interests require the evaluation and refinement of skills This activity helps you develop specific classes and methods that will implement parts of a pokemon game. As you progress through the remainder of the course, you will learn more about the components required to create a full game.

As you complete these tasks, think carefully about the skills you are developing. What components of the pokemon game implementation have been difficult for you to implement? What components seem easy? What can you do to improve your skills so that you can add extensions and additions to the program that go beyond the basic instructions?

Constantly evaluating and refining your skills will allow you to create more complex programs and will allow you to be creative when it's time to implement your final project in the course.

---

1. Remind students to use their textbooks, notes, classroom resources, and online aids to help them answer the questions on WS 5.2.
2. Encourage students to work independently until the last 10 minutes of class.

Students trade work, check, and turn in [10 minutes]

Have students trade and error-check each other's papers. Error-checking partners should write their name on the sheets to share credit for the work.

Accommodation and Differentiation

In classes where reading comprehension is an issue, have students work in pairs today. If you have already created the small group assignments for the next class (see LP 5.3), you can assign pairs that will be in the same group tomorrow.

If you have students who are speeding through this lesson, invite them to create a diagram showing the different parts of a class and instance object that we introduced today. If the diagram is correct and thorough, give the student materials to turn the diagram into a large-format poster for the classroom.

Teacher Prior CS Knowledge

- In *Novice Java Programmers' Conceptions of `Object` and `Class`, and Variation Theory* by Eckerdal and Thuné, novice students view objects at three levels:

  Objects:

  - Object is experienced as a piece of code
  - As above, and in addition object is experienced as something that is active in the program
  - As above, and in addition object is experienced as a model of some real world phenomenon

  Class:

  - Class is experienced as an entity in the program, contributing to the structure of the code
  - As above, and in addition class is experienced as a description of properties and behavior of the object
  - As above, and in addition class is experienced as a description of properties and behavior of the object, as a model of some real world phenomenon

  Software developers are adept at seeing objects and classes as models for both real and abstract constructs.

- For the purposes of the AP test, all fields are private. This ensures good programming practice for beginning students because it forces the use of accessors methods: accessors and mutators. The default access for fields when no access modifier is specified is package private. This means all classes in the same package where the variable is defined can access the field.

Misconceptions

Students add static to class method definitions. Now that Classes have been introduced, static methods that are tied to the class are often confused with non static methods. Students can call static methods without creating an object, however, an object must be created first and the non-static method can be called.

Videos

- BJP 8-2: *Defining a Class* http://media.pearsoncmg.com/aw/aw_reges_bjp_2/videoPlayer.php?id=c8-2
- CSE 142, *Object Oriented Programming State* (19:54–26:25) https://www.youtube.com/watch?v=0IGWknpGPhM&start=1194
- CSE 142, *Object Oriented Programming Behavior* (31:44–42:44) https://www.youtube.com/watch?v=0IGWknpGPhM&start=1905
- CSE 142, *Object methods* (45:32–49:41) https://www.youtube.com/watch?v=0IGWknpGPhM&start=2732

Forum discussion

Lesson 5.02 Object State & Behavior (TEALS Discourse account required)