Learning Objectives - Instance Methods

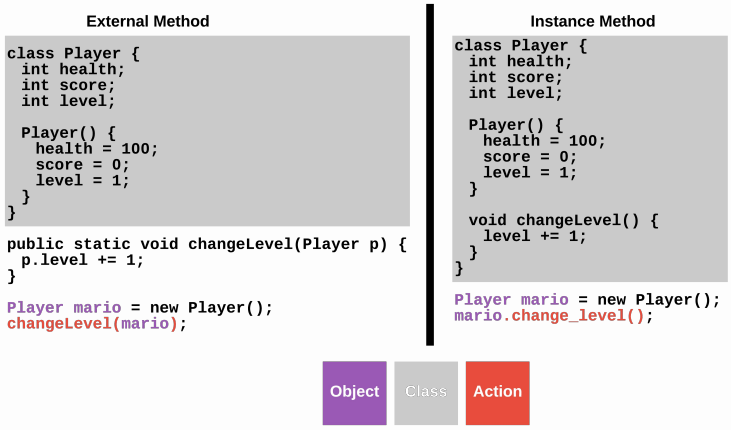
- **Define the term Instance method**
- **Convert an external method that modifies an object into an instance method**
- **Demonstrate the syntax for calling a method**

External Methods vs Class Methods

Class Methods

Back in the introduction to classes and objects lesson, a class was defined as "a collection of data and the actions that can modify the data." The constructor built the "collection of data", but nothing in the class modified the data. Instead, external methods were used to modify the object.

Instead of external methods, instance methods should be used to modify an object. Think of a instance method as a method that is attached to an object. The instance method is the most common type of method when creating classes. Notice how instance methods are declared inside of the class. These methods are called instance methods because they have access to the instance variables (the attributes declared in the constructor). Methods are invoked using dot-notation.



External Methods vs Instance Methods

When mutability was first introduced, you made a `Player` class with a few external methods. You are now going to transform these external methods into instance methods. The `Player` class will be defined just as before. This time, however, `printPlayer` will be a part of the class.

```
//add class definitions below this line

class Player {
  int health;
  int score;
  int level;

  Player() {
    health = 100;
    score = 0;
    level = 1;
  }

  void printPlayer() {
    if (health <= 0) {
      System.out.println("This player is dead. They died on
        level " + level + " with a score of " + score + ".");
    } else {
      System.out.println("This player has " + health + " health,
        a score of " + score + " and is on level " + level +
        ".");
    }
  }
}

//add class definitions above this line
```

Instantiate a `Player` object. Call the class method `printPlayer` using dot-notation.

```
    //add code below this line

    Player mario = new Player();
    mario.printPlayer();

    //add code above this line
```

challenge

Try this variation:

Call `printPlayer` like this:

```
Player mario = new Player();
printPlayer(mario);
```

▼ **Why did this generate an error?**
Java says it cannot find the symbol `printPlayer`, even though the definition is on line 14. Because nothing comes before `printPlayer`, Java assumes that this is an external method. However, `printPlayer` is a part of the `Player` class, which means it is an instance method. Instance methods must be called with dot-notation like `mario.printPlayer();`.

More Player Methods

The next instance methods to add to the `Player` class are those to print the `health` and `level` attributes of the `Player` instance. Start with the instance method `changeHealth`. This method takes `amount` as a parameter. `changeHealth` will add `amount` to the `health` attribute. If a player's health increases, `amount` is positive. If their health decreases, `amount` is negative.

```
void printPlayer() {
  if (health <= 0) {
    System.out.println("This player is dead. They died on
      level " + level + " with a score of " + score + ".");
  } else {
    System.out.println("This player has " + health + " health,
      a score of " + score + " and is on level " + level +
      ".");
  }
}

void changeHealth(int amount) {
  health += amount;
}
```

The instance method `changeLevel` is going to be similar to `changeHealth` except for one difference. `changeLevel` has no parameters. In video games, players go up in levels; rarely do they decrease. So the `level` attribute will increase by one when the instance method is called.

```
void changeHealth(int amount) {
  health += amount;
}

void changeLevel() {
  level += 1;
}
```

challenge

Try these variations:

- Call `changeHealth` and `chagneLevel` for `mario`, and then print the player to make sure the instance methods work.
  ▼ **One possible solution**

```
//add code below this line

Player mario = new Player();
mario.printPlayer();
mario.changeHealth(-10);
mario.changeLevel();
mario.printPlayer();

//add code above this line
```

- Create an instance method to change a player's score?
  ▼ **One possible solution**

```
void changeScore(int amount) {
  score += amount;
}
```

▼ **Why learn about external methods that modify objects when Java has instance methods?**

It might seem like a waste of time to learn how to write external methods that modify objects. But this approach builds upon concepts you have already seen — external methods and objects. This allows you to understand mutability without having to worry about instance methods. Once you understand how these ideas work, transforming an external method into an instance method is much simpler. External methods that

modify objects serve as an intermediary step on the way to learning about instance methods.

More Class Methods

More on Class Methods and Objects

Changes to objects should happen exclusively through instance methods. This makes your code easier to organize and easier for others to understand. Imagine you are going to create a class that keeps track of a meal. In this case, a meal can be thought of as all of the drinks, appetizers, courses, and desserts served. Each one of these categories will become an instance variable (attribute). Assign each attribute an ArrayList of strings.

```
//add class definitions below this line

class Meal {
  ArrayList<String> drinks = new ArrayList<String>();
  ArrayList<String> appetizers = new ArrayList<String>();
  ArrayList<String> mainCourse = new ArrayList<String>();
  ArrayList<String> dessert = new ArrayList<String>();
}

//add class definitions above this line
```

Next, add an instance method to add a drink to the `Meal` object. Use the `.add` method to add an element to the list. So `drinks.add(d)` adds the drink `d` to the ArrayList `drinks`.

```
//add class definitions below this line

class Meal {
  ArrayList<String> drinks = new ArrayList<String>();
  ArrayList<String> appetizers = new ArrayList<String>();
  ArrayList<String> mainCourse = new ArrayList<String>();
  ArrayList<String> dessert = new ArrayList<String>();

  void addDrink(String d) {
    drinks.add(d);
  }
}

//add class definitions above this line
```

Create a `Meal` object and test your code to make sure it is working as expected.

```
  //add code below this line

  Meal dinner = new Meal();
  dinner.addDrink("water");
  System.out.println(dinner.drinks);

  //add code above this line
```

Now create the `addAppetizer` instance method for the class. Like the method above, `addAppetizer` accepts a string as a parameter and adds it to the `appetizers` attribute.

```
void addDrink(String d) {
  drinks.add(d);
}

void addAppetizer(String a) {
  appetizers.add(a);
}
```

Add `"bruschetta"` to the `dinner` object and print it.

```
  //add code below this line

  Meal dinner = new Meal();
  dinner.addDrink("water");
  System.out.println(dinner.drinks);
  dinner.addAppetizer("bruschetta");
  System.out.println(dinner.appetizers);

  //add code above this line
```

challenge

Create the following instance methods:

- `addCourse` - accepts a string which represents a course and adds it to the meal.
- `addDessert` - accepts a string which represents a dessert and adds it to the meal.

Test your code using `"roast chicken"` as a main course and `"chocolate cake"` as a dessert. Then print out each course of the meal.

▼ **Meal code**

```
import java.util.ArrayList;

//add class definitions below this line

class Meal {
  ArrayList<String> drinks = new ArrayList<String>();
  ArrayList<String> appetizers = new ArrayList<String>();
  ArrayList<String> mainCourse = new ArrayList<String>();
  ArrayList<String> dessert = new ArrayList<String>();

  void addDrink(String d) {
    drinks.add(d);
  }

  void addAppetizer(String a) {
    appetizers.add(a);
  }

  void addCourse(String c) {
    mainCourse.add(c);
  }

  void addDessert(String d) {
    dessert.add(d);
  }
}

//add class definitions above this line

public class MoreMethods {
  public static void main(String[] args) {

    //add code below this line

    Meal dinner = new Meal();
    dinner.addDrink("water");
    dinner.addAppetizer("bruschetta");
    dinner.addCourse("roast chicken");
    dinner.addDessert("chocolate cake");

    System.out.println(dinner.drinks);
    System.out.println(dinner.appetizers);
    System.out.println(dinner.mainCourse);
```

```
        System.out.println(dinner.dessert);

        //add code above this line
    }
}
```

Printing the Meal 1

Planning the Method

Before writing the method to print the meal, think about what you want the output should like. Imagine that a meal consists of the following courses:

- Drinks - water and coffee
- Appetizers - nothing served as an appetizer
- Main course - roast chicken, mashed potatoes, and salad.
- Dessert - chocolate cake

Change your code to reflect this meal. Also, add the `printMeal` instance method even though it has not yet been declared.

```
    //add code below this line

    Meal dinner = new Meal();
    dinner.addDrink("water");
    dinner.addDrink("coffee");
    dinner.addCourse("roast chicken");
    dinner.addCourse("mashed potatoes");
    dinner.addCourse("salad");
    dinner.addDessert("chocolate cake");
    dinner.printMeal();

    //add code above this line
```

The `printMeal` instance method is going to invoke the helper method `printCourse`. Call `printCourse` four times, passing it the ArrayList that represents the course as well as the name of the course.

```
void addDessert(String d) {
  dessert.add(d);
}

void printMeal() {
  printCourse(drinks, "drinks");
  printCourse(appetizers, "appetizers");
  printCourse(mainCourse, "main course");
  printCourse(dessert, "dessert");
}
```

The `printCourse` method should be able to handle an empty ArrayList (nothing served), an ArrayList of length 1, an ArrayList of length 2, and an ArrayList of 3 or more elements. Each of these scenarios has specific requirements: Is the verb singular or plural? Do you need a comma-separated list or just the word `"and"`? Should a word be capitalized?

Nothing was Served

Printing a message for an empty ArrayList becomes tricky because the sentence changes based on the course.

- No **drinks were** served with the meal.
- No **appetizers were** served with the meal.
- No **main course was** served with the meal.
- No **dessert was** served with the meal.

Start by checking to see if `course` is an empty ArrayList using the `size` method.

```
void printMeal() {
  printCourse(drinks, "drinks");
  printCourse(appetizers, "appetizers");
  printCourse(mainCourse, "main course");
  printCourse(dessert, "dessert");
}

void printCourse(ArrayList<String> course, String name) {
  if (course.size() == 0) { // check for empty ArrayList

  }
}
```

Next, create a string variable `verb` that will represent the text in bold above. Then ask if the `name` parameter matches each of the four courses: `"drinks"`, `"appetizers"`, `"main course"`, or `"dessert"`. When you have a

match, set `verb` to the appropriate string (the bold text above). Print a sentence that tells the user that no items were served for that course. Be sure to incorporate the variable `verb` to provide the proper context.

```
void printCourse(ArrayList<String> course, String name) {
  if (course.size() == 0) { // check for empty ArrayList
    String verb = "";
    if (name.equals("drinks")) {
      verb = "drinks were";
    } else if (name.equals("appetizers")) {
      verb = "appetizers were";
    } else if (name.equals("main course")) {
      verb = "main course was";
    } else if (name.equals("dessert")) {
      verb = "dessert was";
    }
    System.out.println("No " + verb + " served with the
      meal.");
  }
}
```

Running the code now should produce `"No appetizers were served with the meal."` since it is the only course that is an empty ArrayList.

challenge

Try this variation:

Use the comment symbol `//` to comment out all of the lines with a method that adds a course to the `dinner` object. Run the program. The output should be:

```
No drinks were served with the meal.
No appetizers were served with the meal.
No main course was served with the meal.
No dessert was served with the meal.
```

▼ **Code**

```
    //add code below this line

    Meal dinner = new Meal();
    // dinner.addDrink("water");
    // dinner.addDrink("coffee");
    // dinner.addCourse("roast chicken");
    // dinner.addCourse("mashed potatoes");
    // dinner.addCourse("salad");
    // dinner.addDessert("chocolate cake");
    dinner.printMeal();

    //add code above this line
```

Printing the Meal 2

One Item Was Served

This is a relatively simple case. The trickiest part will be capitalizing the word at the beginning of the sentence. Start by asking if the size of `course` is 1. If only one item is served, that item should be capitalized followed by `" was served with the meal."`.

```
void printCourse(ArrayList<String> course, String name) {
  if (course.size() == 0) { // check for empty ArrayList
    String verb = "";
    if (name.equals("drinks")) {
      verb = "drinks were";
    } else if (name.equals("appetizers")) {
      verb = "appetizers were";
    } else if (name.equals("main course")) {
      verb = "main course was";
    } else if (name.equals("dessert")) {
      verb = "dessert was";
    }
    System.out.println("No " + verb + " served with the
      meal.");
  } else if (course.size() == 1) { // check for one item


  }
}
```

Create the string variable `item` and set it to the first element in `course`. This string needs to be capitalized. Use `substring(0,1)` to represent the first character in the string. Then call the `toUpperCase()` method to capitalize this character. Finally, concatenate this character with the rest of the string, which is represented by `substring(1)`. Print out a sentence using the `item` string.

```
  } else if (course.size() == 1) { // check for one item
    String item = course.get(0);
    item = item.substring(0, 1).toUpperCase() +
      item.substring(1);
    System.out.println(item + " was served with the meal.");
  }
```

Note, remove the comment symbol `//` for `dinner.addDessert("chocolate cake");`.

Two Items Were Served

If there are two items being served, the first item should be capitalized followed by `and` and the second item. The sentence will end with `" were served with the meal."`. Start by asking if the size of `course` is 2.

```
void printCourse(ArrayList<String> course, String name) {
  if (course.size() == 0) {
    String verb = "";
    if (name.equals("drinks")) {
      verb = "drinks were";
    } else if (name.equals("appetizers")) {
      verb = "appetizers were";
    } else if (name.equals("main course")) {
      verb = "main course was";
    } else if (name.equals("dessert")) {
      verb = "dessert was";
    }
    System.out.println("No " + verb + " served with the
      meal.");
  } else if (course.size() == 1) { // check for one item
    String item = course.get(0);
    item = item.substring(0, 1).toUpperCase() +
      item.substring(1);
    System.out.println(item + " was served with the meal.");
  } else if (course.size() == 2) { // check for two items


  }
}
```

Create the string variables `item1` and `item2`. Set them to the two elements in `course`. Capitalize `item1` just as before. Print out a sentence that incorporates `item1` and `item2`.

```
  } else if (course.size() == 2) { // check for two items
    String item1 = course.get(0);
    String item2 = course.get(1);
    item1 = item1.substring(0, 1).toUpperCase() +
      item1.substring(1);
    System.out.println(item1 + " and " + item2 + " were served
      with the meal.");
  }
```

Note, remove the comment symbol `//` for `dinner.addDrink("water");` and `dinner.addDrink("coffee");`.

More than Two Items Were Served

If more than two items are served, then you need a comma-separated list. The first item should be capitalized followed by a comma and a space. The next items are followed by commas and spaces. The final item in the list is prefaced with `and`. No comma is used after the last item. The sentence ends with `" were served with the meal."`. Start by using an `else` statement to capture all instances where the size of `course` is greater than 2.

```
void printCourse(ArrayList<String> course, String name) {
  if (course.size() == 0) {
    String verb = "";
    if (name.equals("drinks")) {
      verb = "drinks were";
    } else if (name.equals("appetizers")) {
      verb = "appetizers were";
    } else if (name.equals("main course")) {
      verb = "main course was";
    } else if (name.equals("dessert")) {
      verb = "dessert was";
    }
    System.out.println("No " + verb + " served with the
      meal.");
  } else if (course.size() == 1) { // check for one item
    String item = course.get(0);
    item = item.substring(0, 1).toUpperCase() +
      item.substring(1);
    System.out.println(item + " was served with the meal.");
  } else if (course.size() == 2) { // check for two items
    String item1 = course.get(0);
    String item2 = course.get(1);
    item1 = item1.substring(0, 1).toUpperCase() +
      item1.substring(1);
    System.out.println(item1 + " and " + item2 + " were served
      with the meal.");
  } else { // more than two items


  }
}
```

Create the string variable `item` and set it to the first element in `course`. Capitalize this string just as before. Use `print` instead of `println` to print this capitalized string followed by a comma and a space.

```
  } else { // more than two items
    String item = course.get(0);
    item = item.substring(0, 1).toUpperCase() +
      item.substring(1);
    System.out.print(item + ", ");
  }
```

Create a for loop to iterate over the `course` ArrayList. We have already printed the first element from the ArrayList. So initialize the loop variable with 1 instead of 0. The last element in `course` needs to have the word `and` appear before element. The last element occurs when `i` is equal to the size of `course` minus 1. Check for this condition, and use a `print` statement when printing `and` and the element. If it is not the last element, use a `print` statement to print the element followed by a comma. After the loop, use a `println` statement to print the rest of the sentence.

```
  } else { // more than two items
    String item1 = course.get(0);
    item1 = item1.substring(0, 1).toUpperCase() +
      item1.substring(1);
    System.out.print(item1 + ", ");
    for (int i = 1; i < course.size(); i++) {
      if (i == course.size() - 1) { // check for last element
        System.out.print("and " + course.get(i) + " ");
      } else {
        System.out.print(course.get(i) + ", ");
      }
    }
    System.out.println("were served with the meal.");
  }
```

Note, remove the comment symbol `//` for remaining lines of code.

challenge

Check your work:

Create different meals and make sure your program works as expected. For example:

```
  //add code below this line

  Meal dinner = new Meal();
  dinner.addDrink("white wine");
  dinner.addAppetizer("tapenade");
  dinner.addAppetizer("antipasto");
  dinner.addCourse("cauliflower bolognese");
  dinner.addCourse("butternut squash soup");
  dinner.addCourse("kale salad");
  dinner.printMeal();

  //add code above this line
```

▼ **Code**

```
import java.util.ArrayList;

//add class definitions below this line

class Meal {
  ArrayList<String> drinks = new ArrayList<String>();
  ArrayList<String> appetizers = new ArrayList<String>();
  ArrayList<String> mainCourse = new ArrayList<String>();
  ArrayList<String> dessert = new ArrayList<String>();

  void addDrink(String d) {
    drinks.add(d);
  }

  void addAppetizer(String a) {
    appetizers.add(a);
  }

  void addCourse(String c) {
    mainCourse.add(c);
  }

  void addDessert(String d) {
    dessert.add(d);
  }

  void printMeal() {
    printCourse(drinks, "drinks");
    printCourse(appetizers, "appetizers");
    printCourse(mainCourse, "main course");
    printCourse(dessert, "dessert");
```

```
  }

  void printCourse(ArrayList<String> course, String name)
      {
    if (course.size() == 0) {
      String verb = "";
      if (name.equals("drinks")) {
        verb = "drinks were";
      } else if (name.equals("appetizers")) {
        verb = "appetizers were";
      } else if (name.equals("main course")) {
        verb = "main course was";
      } else if (name.equals("dessert")) {
        verb = "dessert was";
      }
      System.out.println("No " + verb + " served with the
      meal.");
    } else if (course.size() == 1) { // check for one item
      String item = course.get(0);
      item = item.substring(0, 1).toUpperCase() +
      item.substring(1);
      System.out.println(item + " was served with the
      meal.");
    } else if (course.size() == 2) { // check for two
      items
      String item1 = course.get(0);
      String item2 = course.get(1);
      item1 = item1.substring(0, 1).toUpperCase() +
      item1.substring(1);
      System.out.println(item1 + " and " + item2 + " were
      served with the meal.");
    } else { // more than two items
      String item1 = course.get(0);
      item1 = item1.substring(0, 1).toUpperCase() +
      item1.substring(1);
      System.out.print(item1 + ", ");
      for (int i = 1; i < course.size(); i++) {
        if (i == course.size() - 1) {
          System.out.print("and " + course.get(i) + " ");
        } else {
          System.out.print(course.get(i) + ", ");
        }
      }
      System.out.println("were served with the meal.");
    }
  }
}

//add class definitions above this line

public class MoreMethods {
```

```
  public static void main(String[] args) {

    //add code below this line

    Meal dinner = new Meal();
    dinner.addDrink("white wine");
    dinner.addAppetizer("tapenade");
    dinner.addAppetizer("antipasto");
    dinner.addCourse("cauliflower bolognese");
    dinner.addCourse("butternut squash soup");
    dinner.addCourse("kale salad");
    dinner.printMeal();

    //add code above this line
  }
}
```