Object-Oriented Programming I

Methods – Getters and Setters

Slides by Magdin Stoica
Updates by Georg Feil

Learning Outcomes

1. Define the role and signature and implementation of accessor and mutator methods
2. Create object-oriented programs that use accessor and mutator methods

Reading Assignments

- Introduction to Java Programming (required)
  - Chapter 8, section 8.9: Data Field Encapsulation

- Head First Java (recommended)
  - Chapter 4: Methods Use Instance Variables: How Objects Behave

Accessor and Mutator methods

"Getters and Setters"

Accessor Methods

"Getters"

- A method whose sole purpose is to return the value of a field variable (without changing it) is called an accessor method
  - Accessor methods have the same return type as the field variable whose value they return
  - Accessor methods have no parameters
  - Accessor methods start with the prefix get followed by the Capitalized field name (e.g. getFirstName(), getRadius())
  - Why is this useful?
    - Hint: Fields should normally be private!

Accessor Method Signature

- Accessor method pseudocode

```
public <field var type> get<FieldName>()
{ … }
```

- Java Examples:

```
public String getFirstName() {
    return _firstName;
}

public double getRadius() {
    return _radius;
}
```

Mutator Methods

"Setters"

- A method whose sole purpose is to change the value of a field variable is called a mutator method
  - Mutator methods do not return a value. What is the return type?
  - Mutator methods have one parameter of the same type as the field variable
  - Mutator methods start with the prefix set followed by the Capitalized field name (e.g. setFirstName(…), setRadius(…))

Mutator Method Signature

- Mutator method pseudocode

```
public void set<FieldName>(<field var type> paramName)
{ … }
```

- Java Examples:

```
public void setFirstName(String newName) {
    _firstName = newName;
}

public void setRadius(double radius) {
    _radius = radius;
}
```

Why Use Getters & Setters?

- Using getters and setters allows you to control access to field variables
  - Make all field variables private
  - Add a public accessor (getter) for those fields that need one
  - Add a public mutator (setter) for those fields that need one
  - You can create externally read-only (or write-only) fields, add checking logic to getters/setters etc.

Exercise 1: Using getters and settings

- Examine the program you created for the “barking dogs” exercise (Exercise 4 or 5 in the slides “Inside Classes – Methods” from week 3)
- Can you find a mutator method or “setter”?
  - If it does not follow our rules for mutator methods exactly then fix it so it does
- Add an accessor method (“getter”) for the ‘name’ field
  - Use it to print out the names of the two dogs at the end of main()

Getters & Setters are nothing special

- The compiler treats getter and setter methods just like any other method
  - There's nothing special about them
- Using private field variables along with getters & setters is a programming pattern
  - A good idea or "best practice" that many programmers use in a similar way