

AP Computer Science A

UNIT 2 TOPIC 2 Lab 1
Overloaded Constructors &
Getter/Setter methods



Do Now

1. Warm Up in Google Classroom!

Warm Up!

A student is working with a *Bike* class. The class contains variables to represent the following:

- A string variable called *model* to represent the bike's model
- An int variable called *speed* to represent the bike's speed.

An object named *bicycle* will be declared as type *Bike*.

1. Which one of the following descriptions is accurate?

- A. An attribute of the *model* object is String.
- B. An attribute of the *bicycle* object is *model*.
- C. An attribute of the *Bike* class is *bicycle*.
- D. An instance of the *bicycle* class is *Bike*.
- E. An instance of the *Bike* object is *bicycle*.

2. Pick one of the inaccurate descriptions and rewrite it to make it accurate!


Warm Up!

A student is working with a *Bike* class. The class contains variables to represent the following:

- A string variable called *model* to represent the bike's model
- An int variable called *speed* to represent the bike's speed.

An object named *bicycle* will be declared as type *Bike*.

1. Which one of the following descriptions is accurate?

- 
- A. An attribute of the *model* object is String.
 - B. An attribute of the *bicycle* object is *model*.
 - C. An attribute of the *Bike* class is *bicycle*.
 - D. An instance of the *bicycle* class is *Bike*.
 - E. An instance of the *Bike* object is *bicycle*.

OR: An attribute of the *bicycle* object is *model*
The type of the *model* attribute is String

2. Pick one of the inaccurate descriptions and rewrite it to make it accurate!

Warm Up!

A student is working with a *Bike* class. The class contains variables to represent the following:

- A string variable called *model* to represent the bike's model
- An int variable called *speed* to represent the bike's speed.

An object named *bicycle* will be declared as type *Bike*.

1. Which one of the following descriptions is accurate?

- A. An attribute of the *model* object is String.
- B. An attribute of the *bicycle* object is *model*.
- C. An attribute of the *Bike* class is *bicycle*.
- D. An instance of the *bicycle* class is *Bike*.
- E. An instance of the *Bike* object is *bicycle*.

An instance of the *Bike* class is *bicycle*

OR: An attribute of the *Bike* class is *model* (or *speed*)

2. Pick one of the inaccurate descriptions and rewrite it to make it accurate!


Warm Up!

A student is working with a *Bike* class. The class contains variables to represent the following:

- A string variable called *model* to represent the bike's model
- An int variable called *speed* to represent the bike's speed.

An object named *bicycle* will be declared as type *Bike*.

1. Which one of the following descriptions is accurate?

- A. An attribute of the *model* object is String.
- B. An attribute of the *bicycle* object is *model*.
- C. An attribute of the *Bike* class is *bicycle*.
-  D. An instance of the *bicycle* class is *Bike*.
- E. An instance of the *Bike* object is *bicycle*.

An instance of the *Bike* class is *bicycle*

2. Pick one of the inaccurate descriptions and rewrite it to make it accurate!

Warm Up!

A student is working with a *Bike* class. The class contains variables to represent the following:

- A string variable called *model* to represent the bike's model
- An int variable called *speed* to represent the bike's speed.

An object named *bicycle* will be declared as type *Bike*.

1. Which one of the following descriptions is accurate?

- A. An attribute of the *model* object is String.
- B. An attribute of the *bicycle* object is *model*.
- C. An attribute of the *Bike* class is *bicycle*.
- D. An instance of the *bicycle* class is *Bike*.
- E. An instance of the *Bike* object is *bicycle*.

An instance of the *Bike* class is *bicycle*

2. Pick one of the inaccurate descriptions and rewrite it to make it accurate!

College Board Standards

Unit 2 Topic 1

2.1 Objects: Instances of Classes

5.A Describe the behavior of a given segment of program code.

ENDURING UNDERSTANDING

MOD-1

Some objects or concepts are so frequently represented that programmers can draw upon existing code that has already been tested, enabling them to write solutions more quickly and with a greater degree of confidence.

LEARNING OBJECTIVE

MOD-1.B

Explain the relationship between a class and an object.

ESSENTIAL KNOWLEDGE

MOD-1.B.1

An object is a specific instance of a class with defined attributes.

MOD-1.B.2

A class is the formal implementation, or blueprint, of the attributes and behaviors of an object.

College Board Standards

Unit 2 Topic 2

MOD-1
VAR-1

2.2 Creating and Storing Objects (Instantiation)

1.C Determine code that would be used to interact with completed program code.

3.A Write program code to create objects of a class and call methods.

ENDURING UNDERSTANDING

MOD-1

Some objects or concepts are so frequently represented that programmers can draw upon existing code that has already been tested, enabling them to write solutions more quickly and with a greater degree of confidence.

LEARNING OBJECTIVE

MOD-1.C

Identify, using its signature, the correct constructor being called.

ESSENTIAL KNOWLEDGE

MOD-1.C.1

A signature consists of the constructor name and the parameter list.

MOD-1.C.2

The parameter list, in the header of a constructor, lists the types of the values that are passed and their variable names. These are often referred to as formal parameters.

MOD-1.C.3

A parameter is a value that is passed into a constructor. These are often referred to as actual parameters.

MOD-1.C.4

Constructors are said to be overloaded when there are multiple constructors with the same name but a different signature.

MOD-1.C.5

The actual parameters passed to a constructor must be compatible with the types identified in the formal parameter list.

MOD-1.C.6

Parameters are passed using call by value. Call by value initializes the formal parameters with copies of the actual parameters.

LEARNING OBJECTIVE

MOD-1.D

For creating objects:

- Create objects by calling constructors without parameters.
- Create objects by calling constructors with parameters.

ESSENTIAL KNOWLEDGE

MOD-1.D.1

Every object is created using the keyword `new` followed by a call to one of the class's constructors.

MOD-1.D.2

A class contains constructors that are invoked to create objects. They have the same name as the class.

MOD-1.D.3

Existing classes and class libraries can be utilized as appropriate to create objects.

MOD-1.D.4

Parameters allow values to be passed to the constructor to establish the initial state of the object.

ENDURING UNDERSTANDING

VAR-1

To find specific solutions to generalizable problems, programmers include variables in their code so that the same algorithm runs using different input values.

LEARNING OBJECTIVE

VAR-1.D

Define variables of the correct types to represent reference data.

ESSENTIAL KNOWLEDGE

VAR-1.D.1

The keyword `null` is a special value used to indicate that a reference is not associated with any object.

VAR-1.D.2

The memory associated with a variable of a reference type holds an object reference value or, if there is no object, `null`. This value is the memory address of the referenced object.

Agenda

- Demo: Overloaded constructors & getters/setter methods
- U2T2 Lab 1
- AP Practice Q's in AP Classroom + Google Form Corrections

The keyword `this`

When a parameter name is the *same* as an *instance variable*, we **must** use `this` to refer to the instance variable!

It is quite common to have constructor parameters have the same name as instance variables, so you will see (and do) this a lot!

```
public class Bike {  
  
    // instance variables  
    private String model;  
    private int speed;  
  
    // constructor  
    public Bike(String model, int speed) {  
        this.model = model;  
        this.speed = speed;  
    }  
}
```

"Overloaded" constructors

- When the class has more than one way to create an object!

```
public Bike(String model, int speed) {  
    this.model = model;  
    this.speed = speed;  
}  
  
// another constructor that assigns a default value to speed  
public Bike(String model) {  
    this.model = model;  
    speed = 15; // default value  
}  
  
// a no-parameter constructor that assigns default values to model and speed  
public Bike() {  
    model = "Mountain Bike"; // default value  
    speed = 15; // default value  
}
```

"getter" and "setter" methods

- For accessing and updating an object's instance variables!
- It's hard to fully appreciate why we don't just have instance variables be `public` instead of `private` (because then we could access and update without special methods), but doing so would violate the software engineering premise of **data encapsulation** -- a paradigm that an object should protect its own data (instance variables) and only expose read/write access to outside client code as necessary via getter and setter methods.
- A **"getter"** method is a simple method that returns the value of an object's instance variable (used to "access" or "read" the stored data).
- A **"setter"** method is a simple **void** method that updates the value of an object's instance variable (used to "modify" or change the stored data).

"getter" and "setter" methods

"getter" methods
in the Bike class

```
// "getter" methods, for obtaining values stored in a  
// Bike object's instance variables
```

```
public String getModel() {  
    return model;  
}  
  
public int getSpeed() {  
    return speed;  
}
```

"setter" methods
in the Bike class

```
// "setter" methods, for updating values stored in a  
// Bike object's instance variables
```

```
public void setModel(String newModel) {  
    model = newModel;  
}  
  
public void setSpeed(int newSpeed) {  
    speed = newSpeed;  
}
```


Demo example: Bike class

```
public class Bike {
```

```
// instance variables  
private String model;  
private int speed;
```

instance variables
(data, "attributes")

```
// constructor
```

```
public Bike(String model, int speed) {  
    this.model = model;  
    this.speed = speed;  
}
```

constructors
(overloaded
because more
than one!)

```
// another constructor that assigns a default value to speed
```

```
public Bike(String model) {  
    this.model = model;  
    speed = 15; // default value  
}
```

```
// a no-parameter constructor that assigns default values to model and speed
```

```
public Bike() {  
    model = "Mountain Bike"; // default value  
    speed = 15; // default value  
}
```

```
// "getter" methods, for obtaining values stored in a  
// Bike object's instance variables.
```

```
public String getModel() {  
    return model;  
}
```

```
public int getSpeed() {  
    return speed;  
}
```

"getter"
methods

```
// "setter" methods, for updating values stored in a  
// Bike object's instance variables
```

```
public void setModel(String newModel) {  
    model = newModel;  
}
```

```
public void setSpeed(int newSpeed) {  
    speed = newSpeed;  
}
```

"setter"
methods

```
public String bikeInfo() {  
    return "model: " + model + ", speed: " + speed;  
}
```

all other
methods

Before you leave...

- Make sure you have **shared** your project to GitHub (Git → GitHub → Share Project on GitHub)
 - If you made changes since sharing, be sure to Git → Commit then Git → Push
- Log out of your GitHub account on IntelliJ (File → Settings → Version Control → GitHub → click "-" by your name)
- Open up a **non**-incognito Chrome window, go to github.com, and make sure you are logged out there
- Close your project in IntelliJ (File → Close Project) and remove it from "Recents" (use the gear icon)