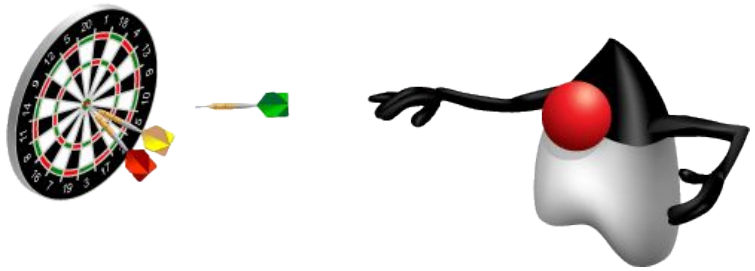
Lesson 5
Working with Objects

Objectives

- After completing this lesson, you should be able to:
  - Declare, instantiate, and initialize object reference variables
  - Compare how object reference variables are stored in relation to primitive variables
  - Access fields on objects
  - Call methods on objects
  - Create a String object
  - Manipulate data by using the String class and its methods
  - Manipulate data by using the StringBuilder class and its methods
  - Use the Java API documentation to explore the methods of a foundation class



Topics

- Declaring, instantiating, and initializing objects
- Working with object references
- Using the String class
- Using the Java API documentation
- Using the StringBuilder class

Working with Objects: Introduction

Objects are accessed via references.

- Objects are instantiated versions of their class.
- Objects consist of attributes and operations:
  - In Java, these are fields and methods.

Accessing Objects by Using a Reference

The camera is like the object that is accessed via the reference (remote).

The remote is like the reference used to access the camera (object).

`Shirt` Class

```
public class Shirt {
  public int shirtID = 0; // Default ID for the shirt
  public String description =
        "-description required-"; // default
  // The color codes are R=Red, B=Blue, G=Green, U=Unset
  public char colorCode = 'U';
  public double price = 0.0; // Default price all items
  // This method displays the details for an item
  public void display() {
    System.out.println("Item ID: " + shirtID);
    System.out.println("Item description:" +
   description);
    System.out.println("Color Code: " + colorCode);
    System.out.println("Item price: " + price);
  } // end of display method
} // end of class
```

Topics

- Declaring, instantiating, and initializing objects
- Working with object references
- Using the String class
- Using the Java API documentation
- Using the StringBuilder class

Working with Object Reference Variables

- Declaration:

```
Classname identifier;
```

Instantiation:

This code fragment creates the object.

```
new Classname();
```

Assignment:

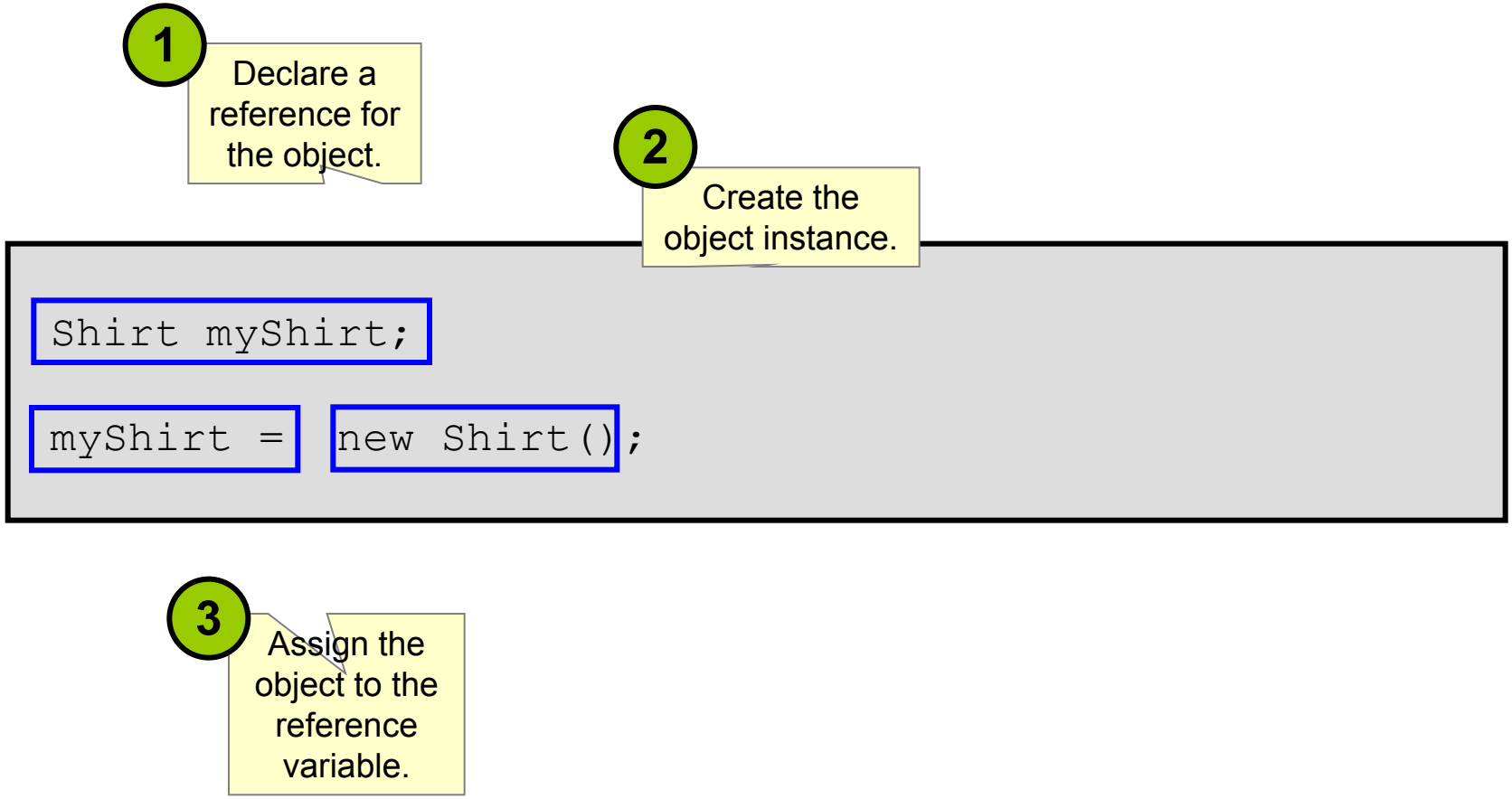
```
Object reference = new Classname();
```

The identifier from the Declaration step

The assignment operator

To assign to a reference, creation and assignment must be in same statement.

Declaring and Initializing: Example



Working with Object References

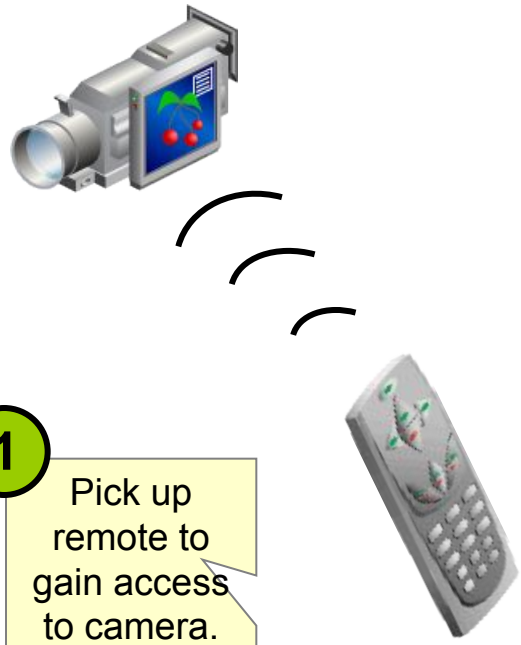
Declare and initialize reference.

```
Shirt myShirt = new Shirt();

int shirtId = myShirt.shirtId;

myShirt.display();
```

Get the value of the `shirtId` field of the object.

Call the `display()` method of the object.

Working with Object References



1. Pick up remote to gain access to camera.

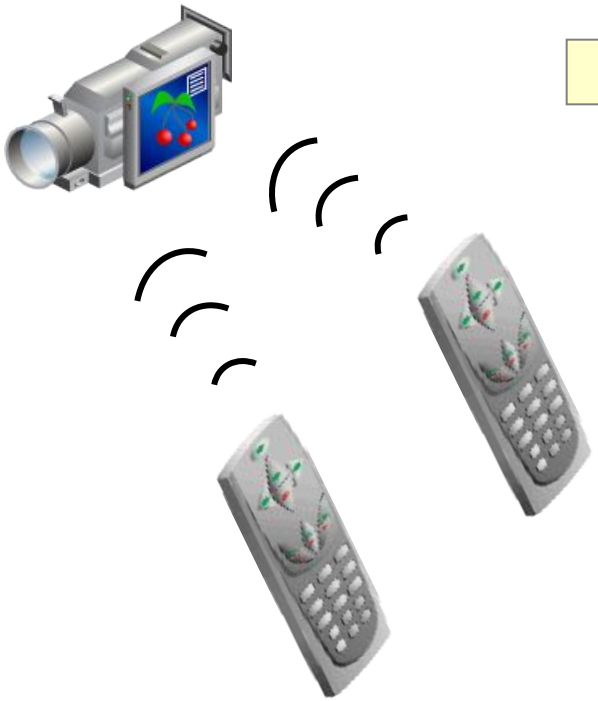
2. Press remote controls to have camera do something.

1. Create a Shirt object and get a reference to it.

```
Shirt myShirt = new Shirt();

myShirt.display();
```

2. Call a method to have Shirt object do something.

Working with Object References



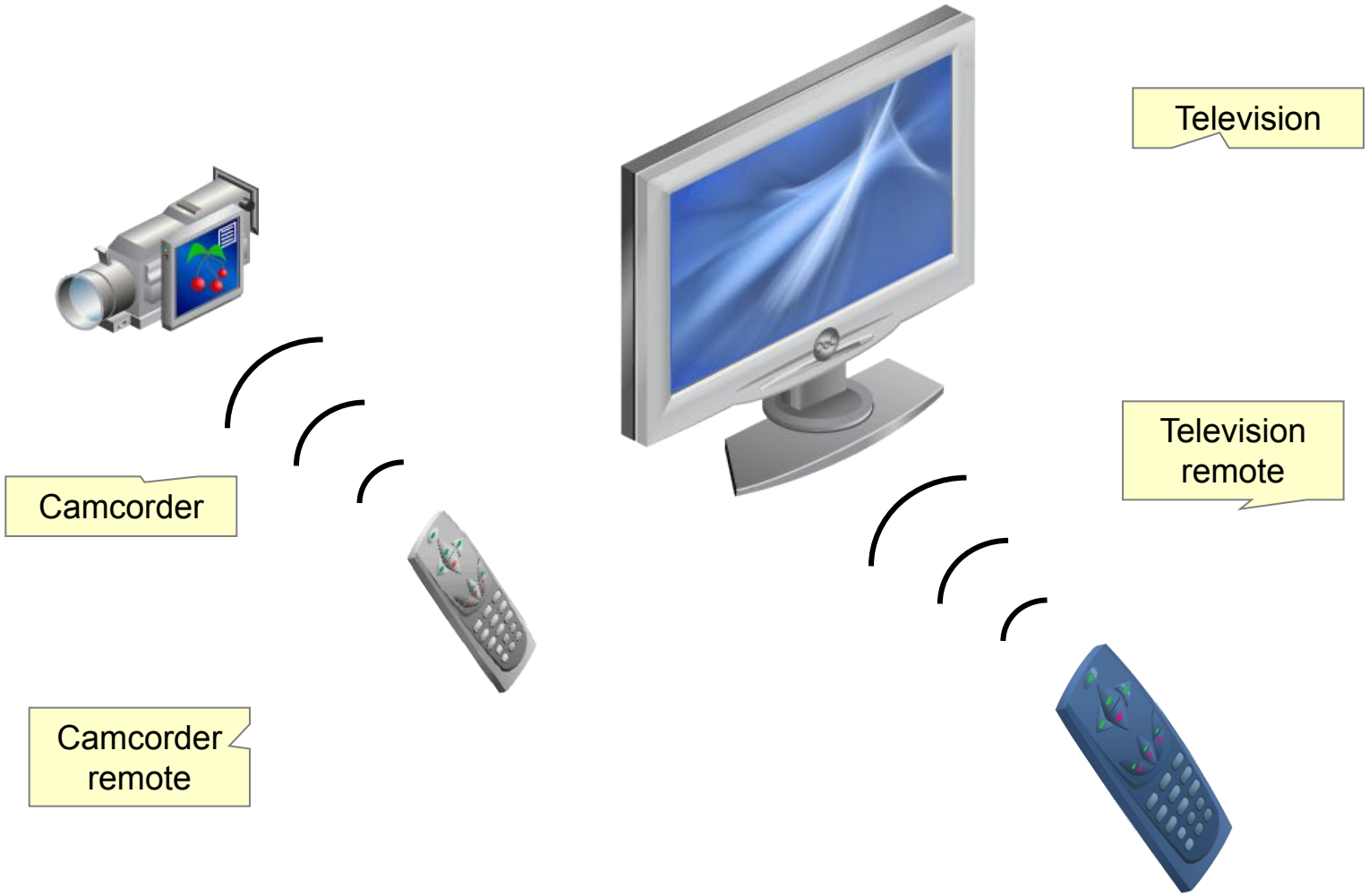
remote2

remote1

There is only one Camera object.

```
Camera remote1 = new Camera();

Camera remote2 = remote1;

remote1.play();

remote2.stop();
```

References to Different Objects



References to Different Object Types

The reference type is Shirt.

The object type is Shirt.

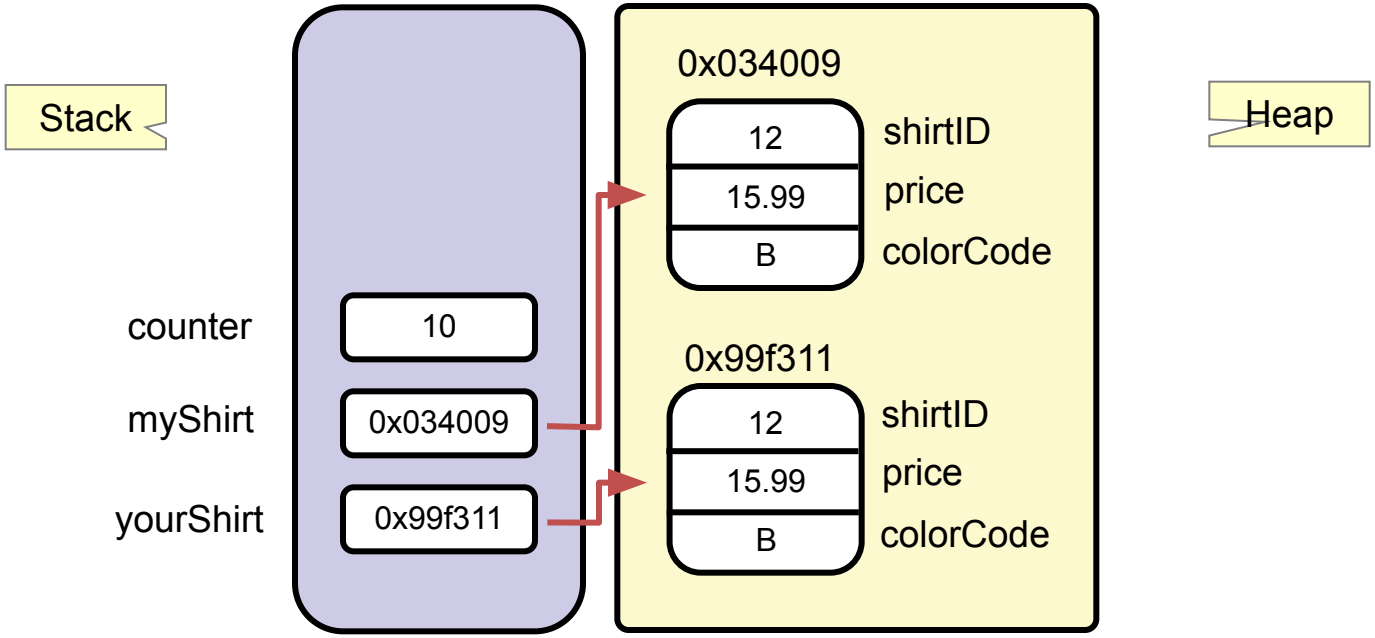
```
Shirt myShirt = new Shirt();
myShirt.display();

Trousers myTrousers = new Trousers();
myTrousers.display();
```

The reference type is Trousers.

The object type is Trousers.

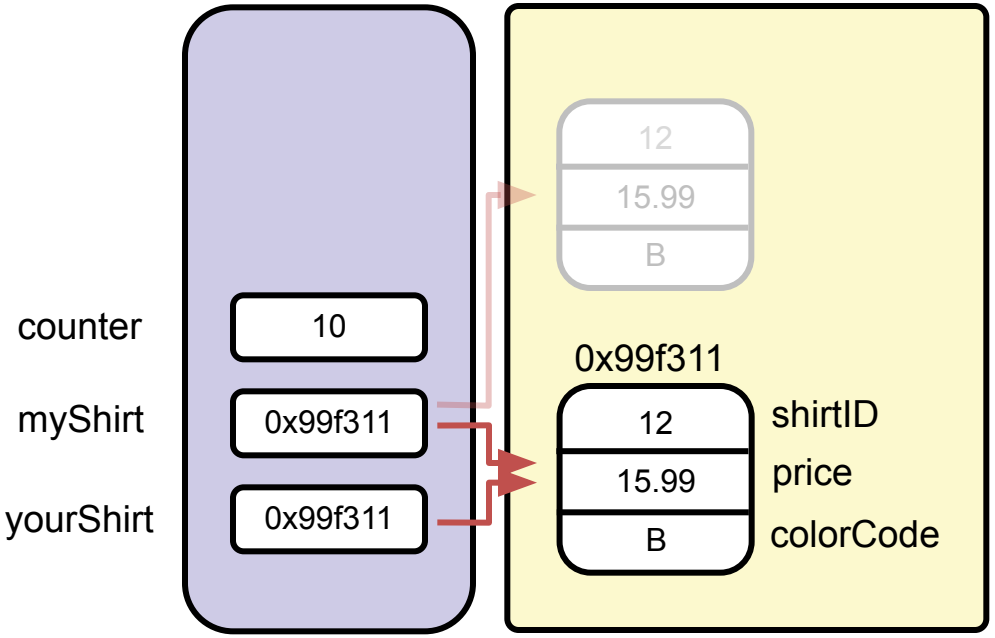
References and Objects In Memory

```
int counter = 10;
Shirt myShirt = new Shirt();
Shirt yourShirt = new Shirt();
```



Assigning a Reference to Another Reference

```
myShirt = yourShirt;
```



Two References, One Object

Code fragment:

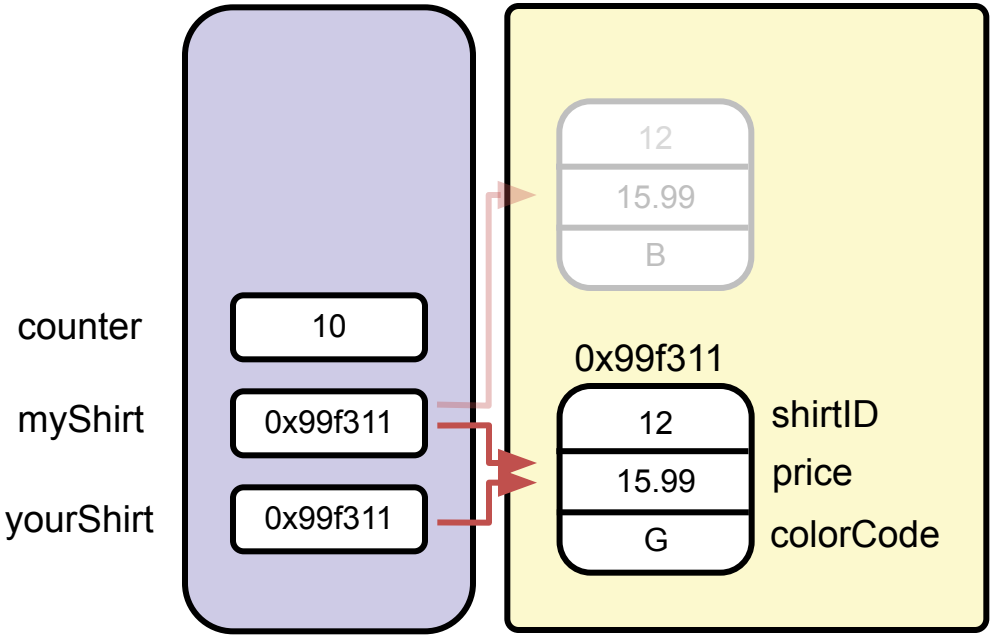
```
Shirt myShirt = new Shirt();
Shirt yourShirt = new Shirt();

myShirt = yourShirt;

myShirt.colorCode = 'R';
yourShirt.colorCode = 'G';

System.out.println("Shirt color: " + myShirt.colorCode);
```

Output from code fragment:

```
 Shirt color: G
```

Assigning a Reference to Another Reference

```
myShirt.colorCode = 'R';
yourShirt.colorCode = 'G';
```



Quiz

Which of the following lines of code instantiates a Boat object and assigns it to a sailBoat object reference?

a. `Boat sailBoat = new Boat();`
b. `Boat sailBoat;`
c. `Boat = new Boat()`
d. `Boat sailBoat = Boat();`

Topics

- Declaring, instantiating, and initializing objects
- Working with object references
- Using the String class
- Using the Java API documentation
- Using the StringBuilder class

String Class

The String class supports some non-standard syntax

- A String object can be instantiated without using the `new` keyword; this is preferred:

```
String hisName = "Fred Smith";
```

  - The `new` keyword can be used, but it is *not* best practice:
    `String herName = new String("Anne Smith");`

- A String object is immutable; its value cannot be changed.
- A String object can be used with the string concatenation operator symbol (+) for concatenation.

Concatenating Strings

When you use a string literal in Java code, it is instantiated and becomes a String reference

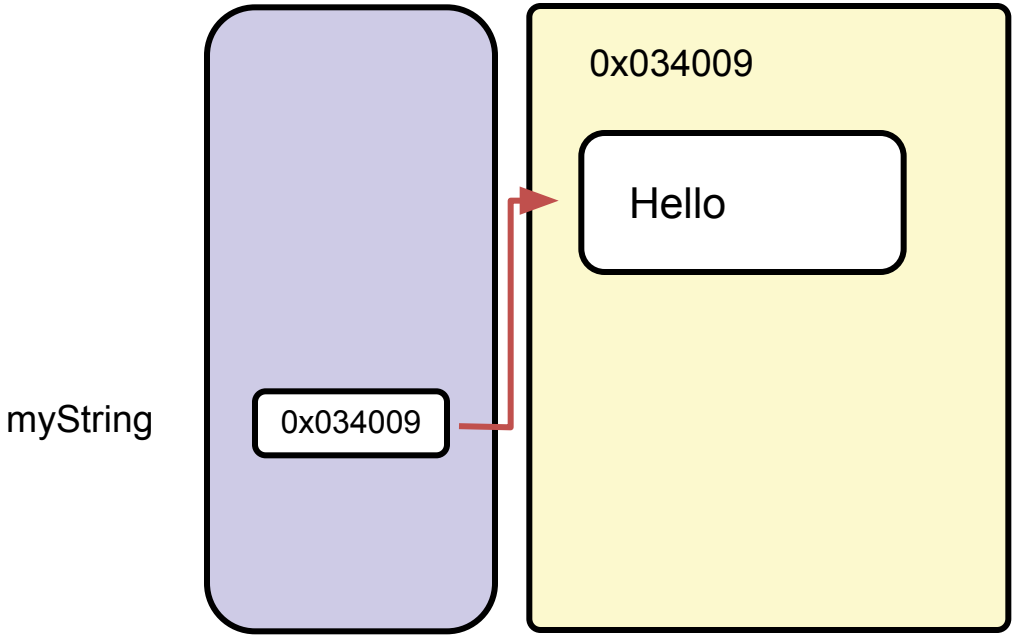
- Concatenate strings:

```
String name1 = "Fred"
theirNames = name1 + " and " +
             "Anne Smith";
```

- The concatenation creates a new string, and the String reference `theirNames` now points to this new string.

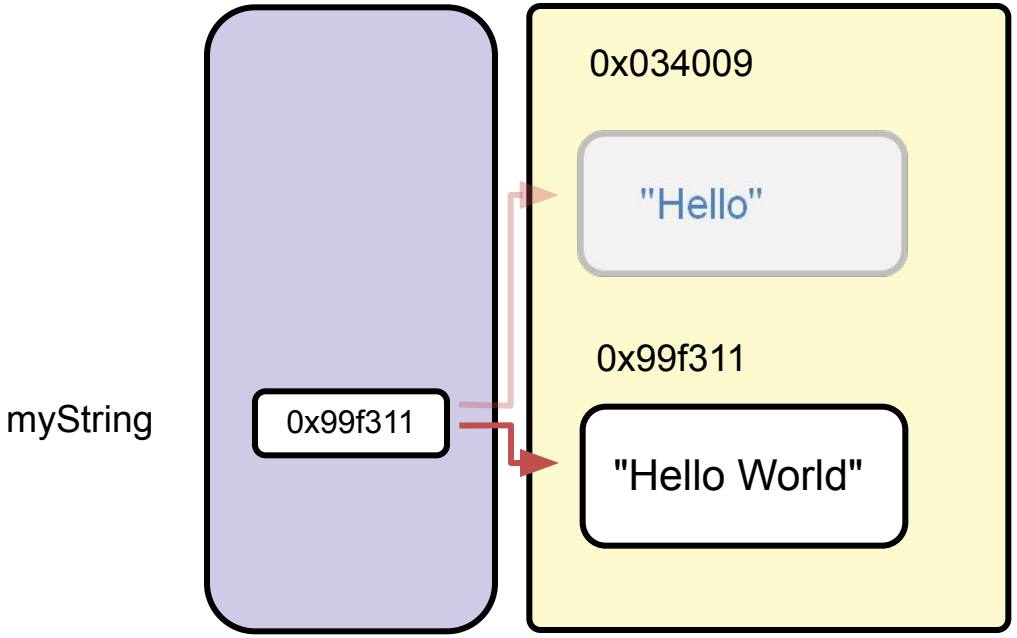
Concatenating Strings

```
String myString = "Hello";
```



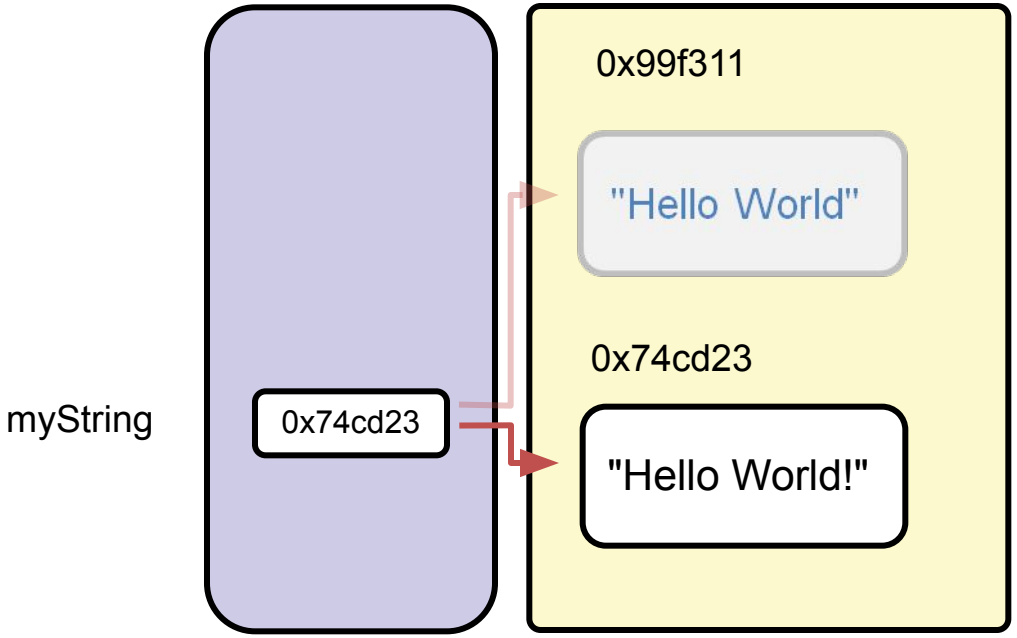
Concatenating Strings

```
String myString = "Hello";
myString = myString.concat(" World");
```



Concatenating Strings

```
String myString = "Hello";
myString = myString.concat(" World");
myString = myString + "!"
```



String Method Calls with Primitive Return Values

A method call can return a single value of any type.

- An example of a method of primitive type `int`:

```
String hello = "Hello World";
int stringLength =
hello.length();
```

String Method Calls with Object Return Values

Method calls returning objects:

```
String greet = " HOW ".trim();

String lc = greet +
"DY".toLowerCase();
```

Or

```
String lc = (greet +
  "DY").toLowerCase();
```

Method Calls Requiring Arguments

- Method calls may require passing one or more arguments:
  - Pass a primitive

```
String theString = "Hello World";
String partString =
theString.substring(6);
```

  - Pass an object

```
boolean endWorld =
            "Hello
World".endsWith("World");
```

Topics

- Declaring, instantiating, and initializing objects
- Working with object references
- Using the String class
- Using the Java API documentation
- Using the StringBuilder class

Java API Documentation

Consists of a set of webpages;

- Lists all the classes in the API
  - Descriptions of what the class does
  - List of constructors, methods, and fields for the class
- Highly hyperlinked to show the interconnections between classes and to facilitate lookup
- Available on the Oracle website at: http://download.oracle.com/javase/7/docs/api/index.html

Java Platform SE 7 Documentation

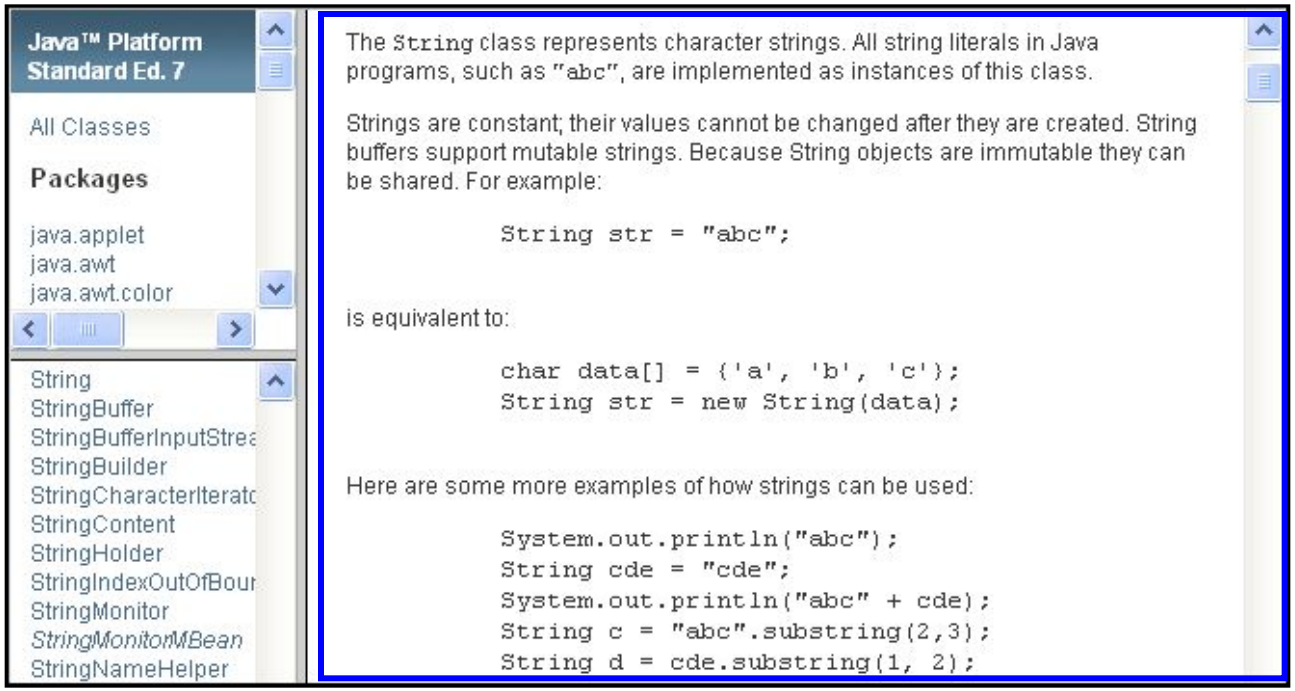
You can select All Classes or a particular package here.

Details about the class selected are shown in this panel.

Depending on what you select, either the classes in a particular package or all classes are listed here.

Java Platform SE 7 Documentation

Scrolling down shows more description of the String class.

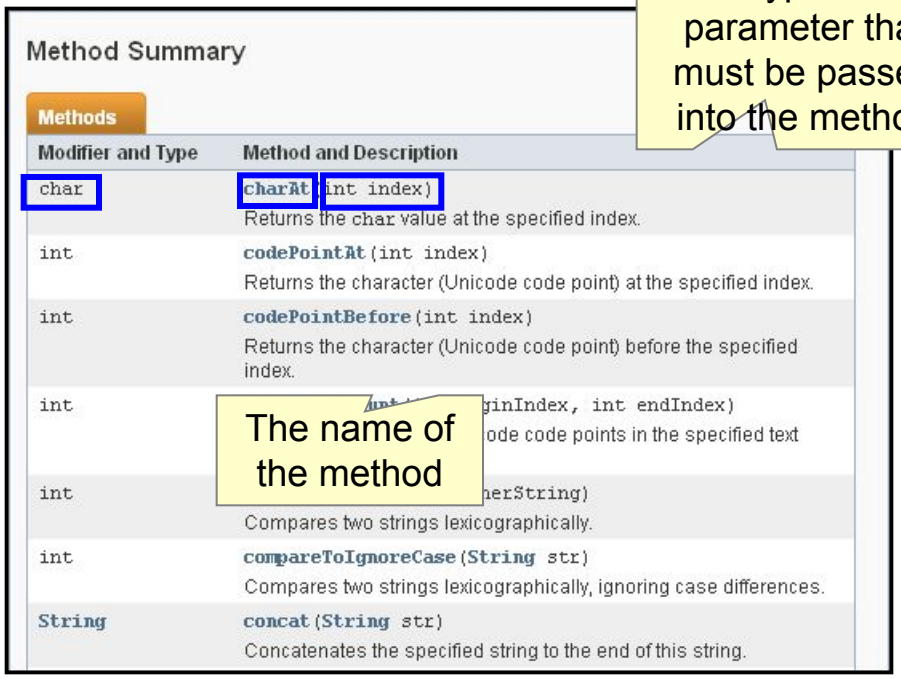


Java Platform SE 7: Method Summary

The type of the method (what type it returns)

The type of the parameter that must be passed into the method

The name of the method



Method Summary

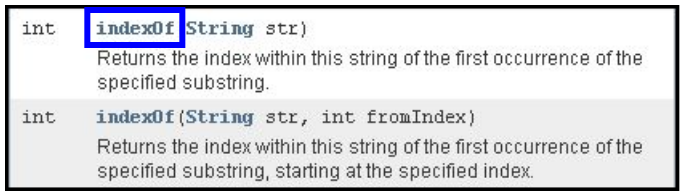
Methods

Modifier and Type	Method and Description
char	charAt(int index) Returns the char value at the specified index.
int	codePointAt(int index) Returns the character (Unicode code point) at the specified index.
int	codePointBefore(int index) Returns the character (Unicode code point) before the specified index.
int	ginIndex, int endIndex) ode code points in the specified text
int	herString) Compares two strings lexicographically.
int	compareToIgnoreCase(String str) Compares two strings lexicographically, ignoring case differences.
String	concat(String str) Concatenates the specified string to the end of this string.

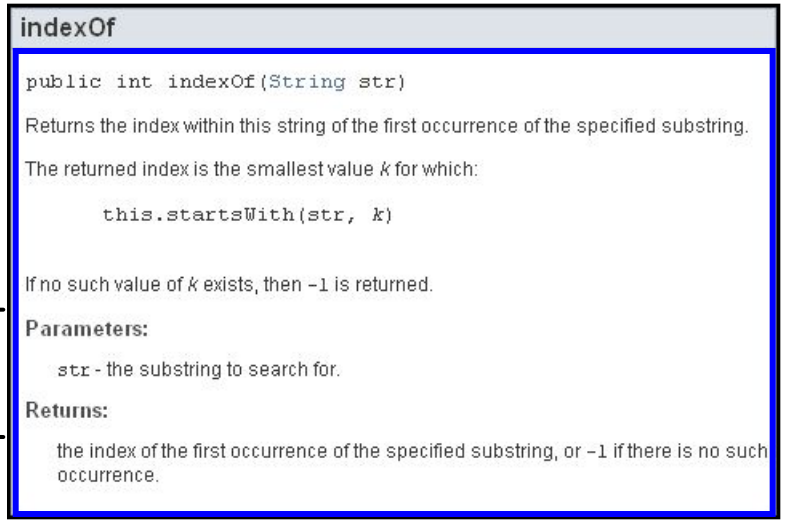
Java Platform SE 7: Method Detail

Click here to get the detailed description of the method.

Detailed description for the `indexOf()` method



Further details about parameters and return value shown in the method list



`System.out` Methods

To find details for `System.out.println()`, consider the following:

- `System` is a class (in `java.lang`).
- `out` is a field of `System`.
- `out` is a reference type that allows calling `println()` on the object type it references.

To find the documentation:

1. Go to `System` class and find the type of the `out` field.
2. Go to the documentation for that field.
3. Review the methods available.

Documentation on `System.out.println()`

java.lang
Class System
java.lang.Object
    java.lang.System

```
public final class System
extends Object
```

Field Summary

Fields

Modifier and Type	Field and Description
static PrintStream	**err** The "standard" error output stream.
static InputStream	**in** The "standard" input stream.
static PrintStream	**out** The "standard" output stream.

void	**print**(Object obj) Prints an object.
void	**print**(String s) Prints a string.
PrintStream	**printf**(Locale l, String format, Object... args) A convenience method to write a formatted string to this output

void	**println**(double x) Prints a double and then terminate the line.
void	**println**(float x) Prints a float and then terminate the line.
void	**println**(int x) Prints an integer and then terminate the line.
void	**println**(long x) Prints a long and then terminate the line.
void	**println**(Object x) Prints an Object and then terminate the line.
void	**println**(String x) Prints a String and then terminate the line.

Some of the methods of `PrintStream`

The field `out` on `System` is of type `PrintStream`.

Using the `print()` and `println()` Methods

The `println` method:

```
System.out.println(data_to_print);
```

Example:

```
System.out.print("Carpe diem ");
System.out.println("Seize the day");
```

This method prints the following:

```
Carpe diem Seize the day
```

Topics

- Declaring, instantiating, and initializing objects
- Working with object references
- Using the String class
- Using the Java API documentation
- Using the StringBuilder class

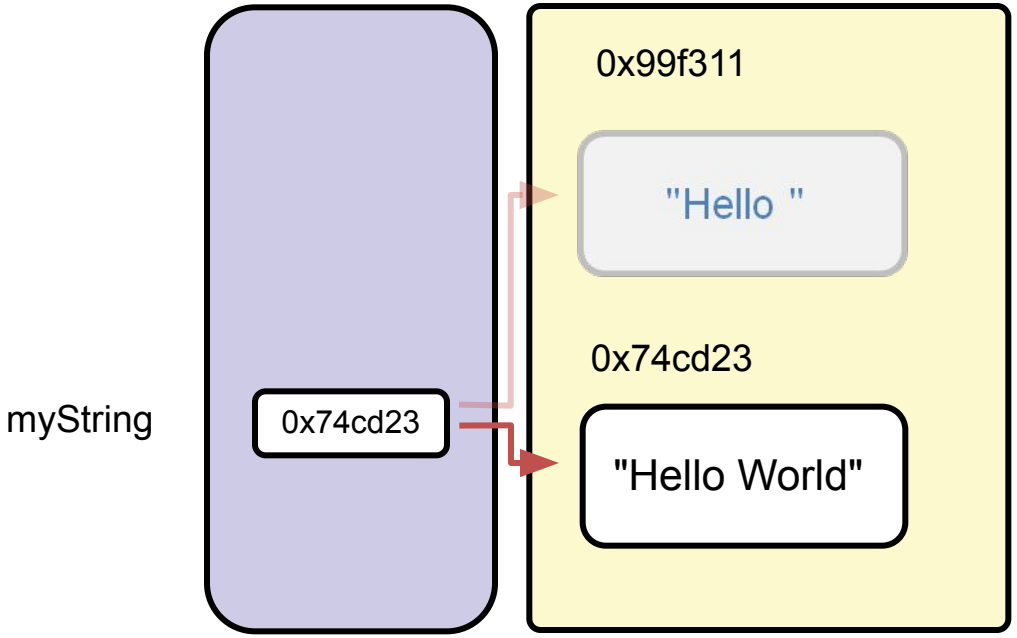
StringBuilder Class

- StringBuilder provides a mutable alternative to String.
- StringBuilder:
  - Is a normal class. Use `new` to instantiate.
  - Has an extensive set of methods for append, insert, delete
  - Has many methods to return reference to current object. There is no instantiation cost.
  - Can be created with an initial capacity best suited to need

  String is still needed because:
  - It may be safer to use an immutable object
  - A class in the API may require a string
  - It has many more methods not available on StringBuilder

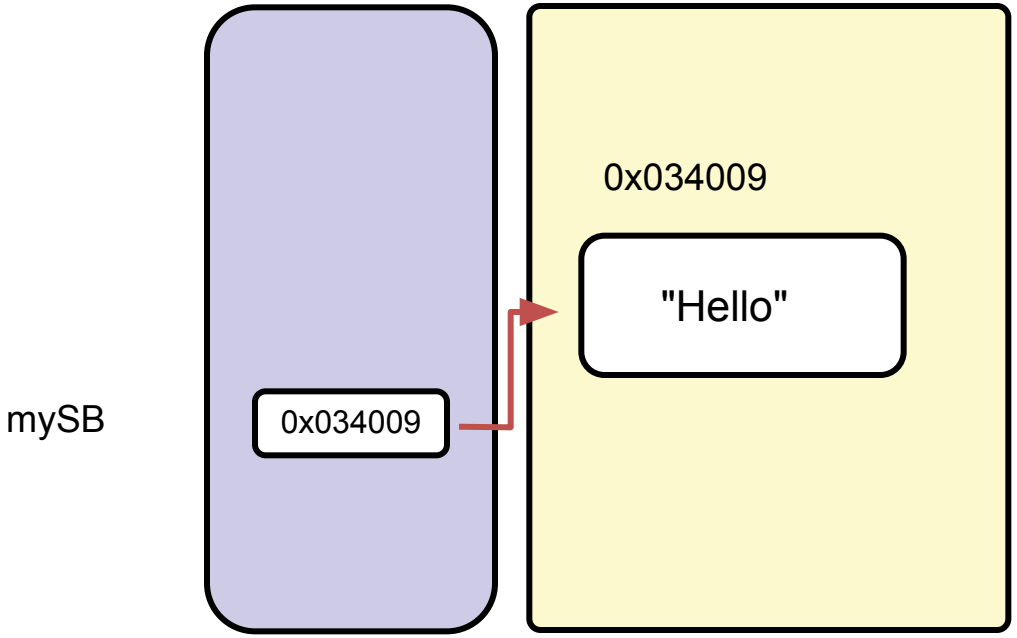
StringBuilder Advantages over String for Concatenation (or Appending)

```
String myString = "Hello";
myString = myString.concat(" World);
```



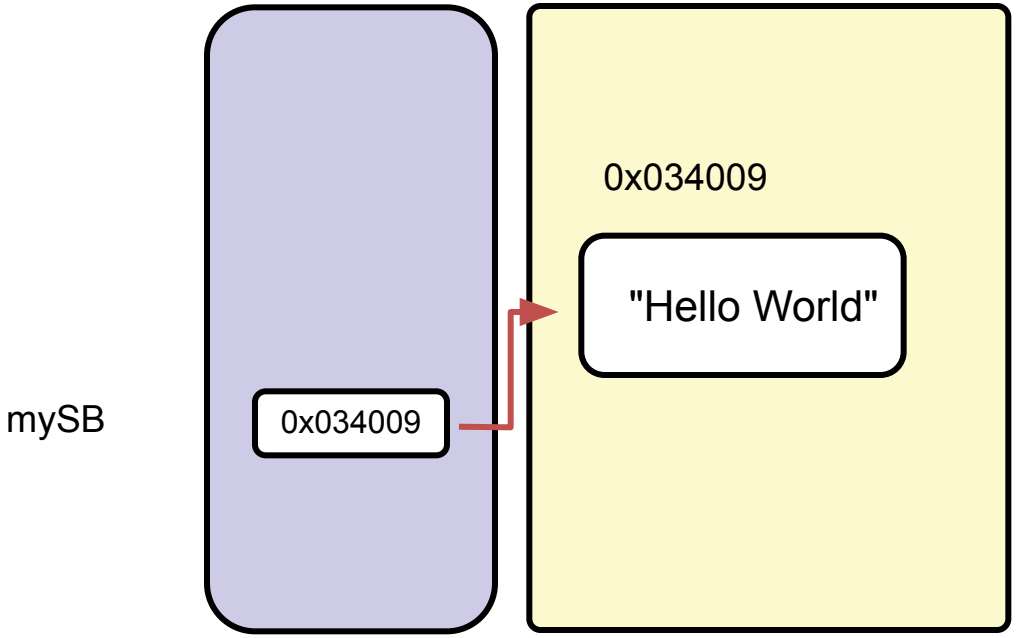
StringBuilder: Declare and Instantiate

```
StringBuilder mySB = new StringBuilder("Hello");
```



StringBuilder Append

```
StringBuilder mySB = new StringBuilder("Hello");
mySB.append(" World");
```



Quiz

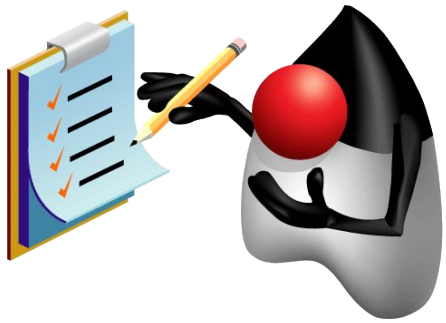
- Which of the following statements are true? (Choose all that apply.)

a. The dot (.) operator creates a new object instance.
b. The String class provides you with the ability to store a sequence of characters.
c. The Java API specification contains documentation for all of the classes in a Java technology product.
d. String objects cannot be modified.

Summary

Objects are accessed via references:

- Objects are instantiated versions of their class.
- Objects consist of attributes and operations:
  - In Java, these are fields and methods.
- To access the fields and methods of an object, get a reference variable to the object:
  - The same object may have more than one reference.
- An existing object's reference can be reassigned to a new reference variable.
- The `new` keyword instantiates a new object and returns a reference.



Practice for Lesson 5 Overview:

- In this practice, you create instances of a class and manipulate these instances in several ways. During the practice, you:
  - Create and initialize object instances
  - Manipulate object references
- In this practice, you create, initialize, and manipulate StringBuilder objects
- In this practice, you examine the Java API specification to become familiar with the documentation and with looking up classes and methods.

  You are not expected to understand everything you see.
  But as you progress through this course, you will understand more and more of the Java API documentation.