AP Computer Science A

UNIT 2 TOPIC 6
Working with String Objects

1. Do Now: Debugging Warm Up!

UNIT 2 Using Objects

College Board Standards
Unit 2 Topic 6

2.6 `String` Objects: Concatenation, Literals, and More

2.A Apply the meaning of specific operators.

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.E

For `String` class:

a. Create `String` objects.

b. Call `String` methods.

ESSENTIAL KNOWLEDGE

VAR-1.E.1

`String` objects can be created by using string literals or by calling the `String` class constructor.

VAR-1.E.2

`String` objects are immutable, meaning that `String` methods do not change the `String` object.

VAR-1.E.3

`String` objects can be concatenated using the + or += operator, resulting in a new `String` object.

VAR-1.E.4

Primitive values can be concatenated with a `String` object. This causes implicit conversion of the values to `String` objects.

VAR-1.E.5

Escape sequences start with a \ and have a special meaning in Java. Escape sequences used in this course include \", \\, and \n.

Do Now: Debugging Warm Up!

One solution:

```
public class Main
{
    public static void main(String[] args)
    {
        GeoLocation geo = new GeoLocation(14, 15);
        geo.printCoords();
    }
}
```

printCoords is a *void* method, so it *cannot* be called "in line" as part of a print statement.

A second solution:

```
public class Main
{
    public static void main(String[] args)
    {
        GeoLocation geo = new GeoLocation(14, 15);
        System.out.println(geo.getCoords());
    }
}
```

getCoords is a non-void method which returns a value (in this case a String), so it *can* be called "in line" as part of a print statement

Mixing types with String Concatenation

```
String a = 1 + 2 + "3";

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

Turn & Talk! Predict what prints and why!

Mixing types with String Concatenation

```
String a = 1 + 2 + "3";

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

Here's what Java does:

```
1 + 2 + "3" → 3 + "3" → "33"  // a string!
```

Mixing types with String Concatenation

```
String a = 1 + 2 + "3";

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

Here's what Java does:

```
1 + 2 + "3" → 3 + "3" → "33"   // a string!
```

Statements are evaluated left to right. So the 1 + 2 happens first as INTEGERS.
Then the integer 3 is concatenated with the String "3" to get "33"

Mixing types with String Concatenation

```
String a = 1 + 2 - "3";

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

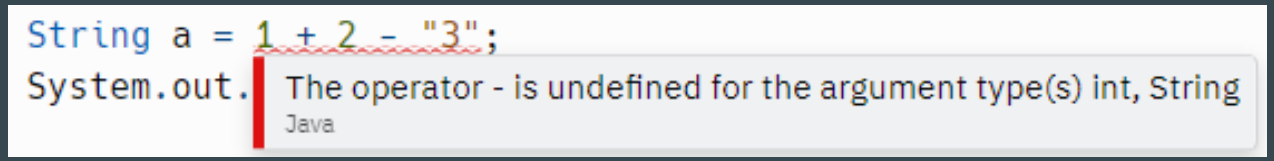
Turn & Talk! Predict what prints and why!

Mixing types with String Concatenation

```
String a = 1 + 2 - "3";

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

Compiler Error!



You cannot use any operator other than "+" on a String!

Strings

- In today's lab, you will learn
  - Different ways to create String objects
  - The subtle rules that govern String concatenation with other primitives (like doubles and ints)
  - "Escape sequences"
  - Strategies for printing or returning multi-line strings

Agenda

1. U2T6 Lab: Working with String Objects
2. Project Fixes (if you missed any points)
3. Tip Calculator (sample solution posted)