AP Computer Science A

UNIT 2 TOPIC 9
Using the Math Class

Do Now!

1. Warm Up problem!

Do Now: Warm Up

a.

```
String a = "WOOT Friday!";
a.toLowerCase();
System.out.println(a);
```

```
WOOT Friday!
```

the `toLowerCase()` method does **NOT** modify the string that it is called on; so **a** is unchanged.

Do Now: Warm Up

a.

```
String a = "WOOT Friday!";
a.toLowerCase();
System.out.println(a);
```

```
WOOT Friday!
```

the `toLowerCase()` method does **NOT** modify the string that it is called on; so **a** is unchanged.

b.

```
String a = "WOOT Friday!";
String b = a.toLowerCase();
System.out.println(a);
System.out.println(b);
```

```
WOOT Friday!
woot friday!
```

the `toLowerCase()` method **returns** a *new* string object comprised of all-lowercase characters of the original string; note here that **a** again remains unchanged, and **b** is the *new* lowercase string that was returned by the method.

Do Now: Warm Up

c.

```
String a = "WOOT Friday!";
a.substring(2, 6);
System.out.println(a);
```

```
WOOT Friday!
```

Similarly, the `substring()` method does **NOT** modify the string that it is called on; so **a** is unchanged. In other words, it does *not* "chop out" a substring from the original string in any way!

Do Now: Warm Up

c.

```
String a = "WOOT Friday!";
a.substring(2, 6);
System.out.println(a);
```

```
WOOT Friday!
```

Similarly, the `substring()` method does **NOT** modify the string that it is called on; so **a** is unchanged. In other words, it does *not* "chop out" a substring from the original string in any way!

d.

```
String a = "WOOT Friday!";
String b = a.substring(2, 6);
System.out.println(a);
System.out.println(b);
```

```
WOOT Friday!
OT F
```

The `substring()` method **returns** a *new* string object comprised of the specified characters of the original string; note here that **a** again remains unchanged, and **b** is the *new* "substring" that was returned by the method.

Debugging Java Errors

You likely saw at least one `IndexOutOfBoundsException` while writing your code yesterday!

```
"C:\Program Files\Java\jdk-17.0.1\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community
Halloeen
alloween
Hallowee
Exception in thread "main" java.lang.StringIndexOutOfBoundsException Create breakpoint : begin 10, end 9, length 9
    at java.base/java.lang.String.checkBoundsBeginEnd(String.java:4601)
    at java.base/java.lang.String.substring(String.java:2704)
    at java.base/java.lang.String.substring(String.java:2677)
    at CustomStringMethods.removeCharacter(CustomStringMethods.java:172)
    at Main.main(Main.java:10)

Process finished with exit code 1
```

Debugging Java Errors

You likely saw at least one `IndexOutOfBoundsException` while writing your code yesterday!

```
"C:\Program Files\Java\jdk-17.0.1\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community
Halloeen
alloween
Hallowee
Exception in thread "main" java.lang.StringIndexOutOfBoundsException Create breakpoint : begin 10, end 9, length 9
    at java.base/java.lang.String.checkBoundsBeginEnd(String.java:4601)
    at java.base/java.lang.String.substring(String.java:2704)
    at java.base/java.lang.String.substring(String.java:2677)
    at CustomStringMethods.removeCharacter(CustomStringMethods.java:172)
    at Main.main(Main.java:10)

Process finished with exit code 1
```

This is a **runtime** error, since clearly the program had started running!

Debugging Java Errors

You likely saw at least one `IndexOutOfBoundsException` while writing your code yesterday!

```
"C:\Program Files\Java\jdk-17.0.1\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community
Halloeen
alloween
Hallowee
Exception in thread "main" java.lang.StringIndexOutOfBoundsException Create breakpoint : begin 10, end 9, length 9
    at java.base/java.lang.String.checkBoundsBeginEnd(String.java:4601)
    at java.base/java.lang.String.substring(String.java:2704)
    at java.base/java.lang.String.substring(String.java:2677)
    at CustomStringMethods.removeCharacter(CustomStringMethods.java:172)
    at Main.main(Main.java:10)

Process finished with exit code 1
```

It's also an **exception**, which are *always* runtime errors!

Debugging Java Errors

You likely saw at least one `IndexOutOfBoundsException` while writing your code yesterday!

```
"C:\Program Files\Java\jdk-17.0.1\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community
Halloeen
alloween
Hallowee
Exception in thread "main" java.lang.StringIndexOutOfBoundsException Create breakpoint : begin 10, end 9, length 9
    at java.base/java.lang.String.checkBoundsBeginEnd(String.java:4601)
    at java.base/java.lang.String.substring(String.java:2704)
    at java.base/java.lang.String.substring(String.java:2677)
    at CustomStringMethods.removeCharacter(CustomStringMethods.java:172)
    at Main.main(Main.java:10)

Process finished with exit code 1
```

here is the "**method call stack**" when the exception occurred

Debugging Java Errors

You likely saw at least one `IndexOutOfBoundsException` while writing your code yesterday!

```
"C:\Program Files\Java\jdk-17.0.1\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community
Halloeen
alloween
Hallowee
Exception in thread "main" java.lang.StringIndexOutOfBoundsException Create breakpoint : begin 10, end 9, length 9
    at java.base/java.lang.String.checkBoundsBeginEnd(String.java:4601)
    at java.base/java.lang.String.substring(String.java:2704)
    at java.base/java.lang.String.substring(String.java:2677)
    at CustomStringMethods.removeCharacter(CustomStringMethods.java:172)
    at Main.main(Main.java:10)

Process finished with exit code 1
```

to debug: start at the bottom and work your way up *until you reach a method whose code you can't see* (in this case, we can't see the java.lang.String code)

Debugging Java Errors

You likely saw at least one `IndexOutOfBoundsException` while writing your code yesterday!

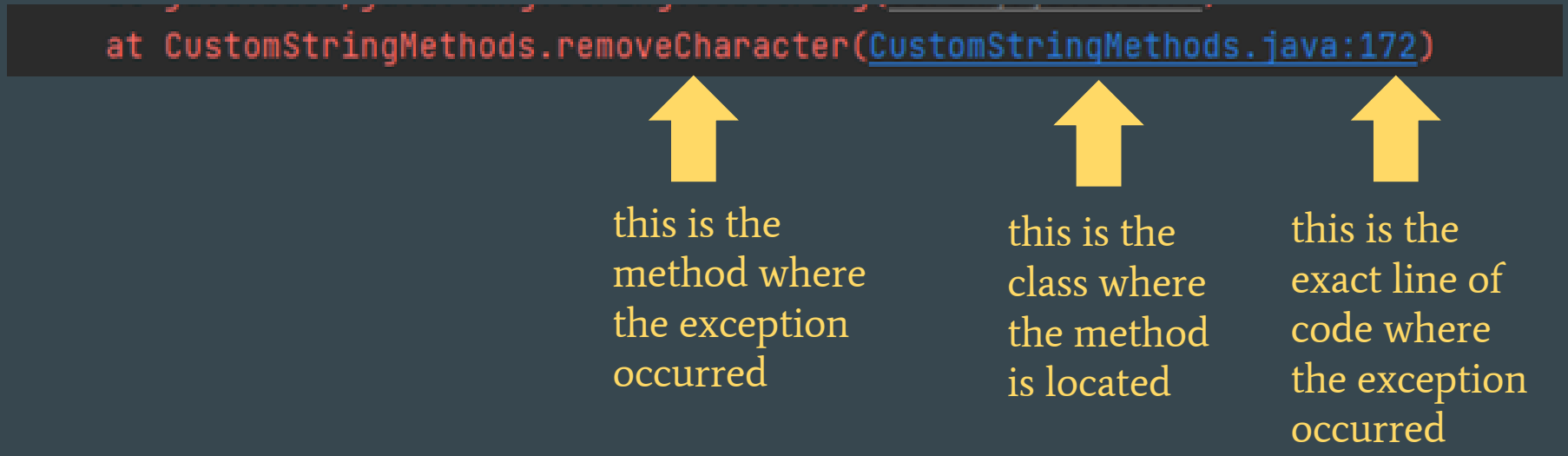
```
"C:\Program Files\Java\jdk-17.0.1\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community
Halloeen
alloween
Hallowee
Exception in thread "main" java.lang.StringIndexOutOfBoundsException Create breakpoint : begin 10, end 9, length 9
    at java.base/java.lang.String.checkBoundsBeginEnd(String.java:4601)
    at java.base/java.lang.String.substring(String.java:2704)
    at java.base/java.lang.String.substring(String.java:2677)
    at CustomStringMethods.removeCharacter(CustomStringMethods.java:172)
    at Main.main(Main.java:10)

Process finished with exit code 1
```

the last method whose code you can read is where you should look!

Debugging Java Errors

You likely saw at least one `IndexOutOfBoundsException` while writing your code yesterday!



UNIT 2 Using Objects

College Board Standards

Unit 2 Topic 9

2.9 Using the `Math` Class

1.B Determine code that would be used to complete code segments.

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	ESSENTIAL KNOWLEDGE
MOD-1.H Call static methods.	MOD-1.H.1 Static methods are called using the dot operator along with the class name unless they are defined in the enclosing class.

ENDURING UNDERSTANDING

CON-1

The way variables and operators are sequenced and combined in an expression determines the computed result.

LEARNING OBJECTIVE	ESSENTIAL KNOWLEDGE
CON-1.D Evaluate expressions that use the `Math` class methods.	CON-1.D.1 The `Math` class is part of the `java.lang` package. CON-1.D.2 The `Math` class contains only static methods.

LEARNING OBJECTIVE	ESSENTIAL KNOWLEDGE
CON-1.D Evaluate expressions that use the `Math` class methods.	CON-1.D.3 The following static `Math` methods—including what they do and when they are used—are part of the Java Quick Reference: ▪ `int abs(int x)` — Returns the absolute value of an `int` value ▪ `double abs(double x)` — Returns the absolute value of a `double` value ▪ `double pow(double base, double exponent)` — Returns the value of the first parameter raised to the power of the second parameter ▪ `double sqrt(double x)` — Returns the positive square root of a `double` value ▪ `double random()` — Returns a double value greater than or equal to 0.0 and less than 1.0 CON-1.D.4 The values returned from `Math.random` can be manipulated to produce a random `int` or `double` in a defined range.

Math Class Methods

Instance methods vs. Static methods

- All the methods we have designed so far have been **instance methods**, which means you invoke them by first creating an object of that class (with the new keyword), then using dot notation on that **object** (which is an **instance** of the class)

- **Example:** the `length`, `substring`, `indexOf`, `compare`, and `equals` methods are all **instance methods** in the String class:

```
String name = "Frank";                          // creating a string
object
int len = name.length();                  // method invoked on the object
String sub = name.substring(1, 3); // method invoked on the object
int idx = name.indexOf("a");          // method invoked on the object
String sub = name.equals("Frank"); // method invoked on the object
```

Instance methods vs. Static methods

- All the methods we have designed so far have been **instance methods**, which means you invoke them by first creating an object of that class (with the new keyword), then using dot notation on that **object** (which is an **instance** of the class)

- **Example:** the `length`, `substring`, `indexOf`, `compare`, and `equals` methods are all **instance methods** in the String class:

```
String name = "Frank";                          // creating a string
object
int len = name.length();                  // method invoked on the object
String sub = name.substring(1, 3); // method invoked on the object
int idx = name.indexOf("a");          // method invoked on the object
String sub = name.equals("Frank"); // method invoked on the object
```

- **Example:** the `printArea()` method on our `Rectangle` class:

```
Rectangle rect = new Rectangle(4, 8); // creating the object
int len = rect.printArea();           // method invoked on the object
```

Instance methods vs. Static methods

- But we can create **static methods** in a class, which are invoked using the **class name** followed by dot notation -- they are **NOT** invoked on instances (objects) of the class

- **Example:** the `valueOf` method of the `String` class is a **static** method of the String class; we don't first create a string object, but rather call it *on the class itself:*

```
String intAsStr = String.valueOf(5);  // invoked on the Class
```

Instance methods vs. Static methods

- But we can create **static methods** in a class, which are invoked using the **class name** followed by dot notation -- they are **NOT** invoked on instances (objects) of the class

- **Example:** the `valueOf` method of the `String` class is a **static** method of the String class; we don't first create a string object, but rather call it *on the class itself:*

```
String intAsStr = String.valueOf(5);  // invoked on the Class
```

- We will discuss the answers to "*What are static methods? And when should a method in a class be made static?*" in Unit 5 when we discuss class design.

- **For now: You just need to know that static methods exist, and how to call a static method (on the class, not an object).**

Static Variables

- Relatedly, we have used **static variables** before; these are *variables* that are accessed on the **class**, *not* specific instances of a class (these are the ***instance* variables**!)

Static Variables

- Relatedly, we have used **static variables** before; these are *variables* that are accessed on the **class**, *not* specific instances of a class (these are the ***instance* variables**!)

- Just like static methods, to access static variables, we use the class name followed by dot notation; since it's not a method, so we don't use `()`

Static Variables

- Relatedly, we have used **static variables** before; these are *variables* that are accessed on the **class**, *not* specific instances of a class (these are the ***instance* variables**!)

- Just like static methods, to access static variables, we use the class name followed by dot notation; since it's not a method, so we don't use `()`

- **`Integer.MIN_VALUE`**
  `Integer.MAX_VALUE`
  `Math.PI` `← a new one!`

- Note that these variable names are all capitalized because they are *also* constants (in other words, they are **static *final*** variables) -- but static variables do not need to also need to be constants, and we will see how this all works in Unit 5.

Static Variables

- Relatedly, we have used **static variables** before; these are *variables* that are accessed on the **class**, *not* specific instances of a class (these are the ***instance* variables**!)

- Just like static methods, to access static variables, we use the class name followed by dot notation; since it's not a method, so we don't use `()`

- **`Integer.MIN_VALUE`**
  `Integer.MAX_VALUE`
  `Math.PI` `← a new one!`

- Note that these variable names are all capitalized because they are *also* constants (in other words, they are **static *final*** variables) -- but static variables do not need to also need to be constants, and we will see how this all works in Unit 5.

- **For now: You just need to know that static variables exist, and how to access and use the ones shown above!**

Math Class Methods to Know

Java Quick Reference

Accessible methods from the Java library that may be included in the exam

Class Constructors and Methods	Explanation
Math Class	
`static int abs(int x)`	Returns the absolute value of an `int` value
`static double abs(double x)`	Returns the absolute value of a `double` value
`static double pow(double base, double exponent)`	Returns the value of the first parameter raised to the power of the second parameter
`static double sqrt(double x)`	Returns the positive square root of a `double` value
`static double random()`	Returns a `double` value greater than or equal to `0.0` and less than `1.0`
Integer Class	
`Integer(int value)`	Constructs a new `Integer` object that represents the specified `int` value
`Integer.MIN_VALUE`	The minimum value represented by an `int` or `Integer`
`Integer.MAX_VALUE`	The maximum value represented by an `int` or `Integer`
`int intValue()`	Returns the value of this `Integer` as an `int`

These are the FIVE Math methods to know; they are all static methods, which means you call them with: `Math.methodName`

`Math.PI` is very good to know :)

`Math.PI` is not on the AP Exam, but you should know it :) → Returns the value of pi

These are the TWO constants to know (both static variables)

```
int abs1 = Math.abs(-5);  // version of abs that returns an int
System.out.println(abs1);

double abs2 = Math.abs(-5.8); // version of abs that returns a double
System.out.println(abs2);

double power = Math.pow(2, 3);
System.out.println(power);

double root = Math.sqrt(30);
System.out.println(root);

double randNum = Math.random(); // no parameter; produces double: 0.0 <= num < 1.0
System.out.println(randNum);

// static constant variables to know
System.out.println(Math.PI);
System.out.println(Integer.MIN_VALUE);
System.out.println(Integer.MAX_VALUE);
```

Examples of each

Notes!

These are pretty easy methods to use; the biggest challenge is differentiating *how* to use them, as they are *static* methods (called on the class), versus all the methods we have been calling so far which have been *instance* methods (called on *objects* of the class).

Note we don't need to import Math class... any ideas why?

Notes!

These are pretty easy methods to use; the biggest challenge is differentiating *how* to use them, as they are *static* methods (called on the class), versus all the methods we have been calling so far which have been *instance* methods (called on *objects* of the class).

Note we don't need to import Math class... any ideas why? It's part of the `java.lang` library, which gets imported automatically (along with String and System).

Agenda

- U2T9 Lab: Using the Math class
- U2T9 AP Practice Q's in AP Classroom (3 Q's) + Google Form

Exit Slip

```
int rand = (int) (Math.random() * (max - min + 1)) + min
```

```
int rand = (int) (Math.random() * (75 - 50 + 1)) + 50
```

```
int rand = (int) (Math.random() * 26) + 50
```