Intended Audience and Course

This lesson should be appropriate for any high school student’s first computer science class centered around written code. I imagine that means that this lesson would mostly be used in intro computer science classes, but I am aware that some intro CS classes start with block coding like Scratch or Snap, meaning this might be useful for classes that come after that. The materials for this lesson are very much centered around Java as programming language, more so than lesson 00 gets into how Java specifically handles errors. This could fit well in an AP CS A course but it could be applied to other languages as well with enough modifications as the actual core of the lesson is not a Java exclusive idea.

Explanation of Methods Used and Lesson Goals

Although as a class we would probably learn about errors and debugging naturally as the course went on, I feel like it would probably be good to devote at least one lesson to talking about errors. More specifically, the goal of this lesson would be for students to identify and differentiate between three types of programming errors; compilation errors, runtime errors, and logical errors. This lesson would probably come after both arrays/lists and loops have been introduced, as arrays can be a common cause of runtime errors and loops can be a common cause of logical errors. In a sense, this lesson could also serve as a refresher/review of both concepts. I'm unsure if this would be able to fit in this one lesson or if it would necessitate its own, but it would also be a good place for a class discussion of what errors students have noticed come up in their programming experience so far, and what has been their ways of dealing with them.

There are a lot of methods that could potentially be employed for such a lesson. For example, this could potentially include live coding to demonstrate what different errors in a program might look like when run. Students could have scaffolded but error-prone code as an activity where they have to work on both identifying and fixing errors in the code they've been presented. I've also thought of a potential group activity where each student has a chance to write, sabotage, and debug code given to them by other members of their group.

Lesson Structure

Aim: How can we identify and differentiate between the three types of programming errors?

Warm-Up (3-5 minutes)

Do Now: What is the difference between these two programs and the errors they produce? How would we fix both programs?

Students will be given 3-5 minutes to answer these questions in their notebook (or something equivalent). A useful hint for students who might be struggling is to look carefully at what the error messages say.

Warm-Up Review (3 - 5 minutes)

Although the Do Now uses pictures of the two different programs, we'll go over the Do Now by using live coding (provided in materials as main), using the code with syntax errors as the starting point

The review of the Do Now should follow a sequence somewhat like this. Bullets in quotes are examples of what you might ask the class

1.
    - Ask the class a similar question to the Do Now, "what is the difference between the two console outputs?"
    - Students wordings may vary, but it should be something along the lines of "one compiles while the other doesn't" or "They have different error messages"
2.
    - If it hasn't been pointed out yet already, point out that the two different code samples have two different error messages
    - "Now that we know what the errors are, can we find what in the code/where in the code these errors are being caused by?"
    - If a student doesn't point it out, point out that the error messages will tend to have line numbers next to them to help find the errors
3.
    - "Now that we've identified what the errors are and where they are in the code, what should we do to fix these errors?"
    - Apply the coding fixes live
4.
    - "Now that we've fixed these errors, let's try running this program to see if it works"
    - Run the program and show that it compiles and runs, but the output is different than what would be expected
    - "We don't get any error messages when we run our code, but is this output what we'd expect?"
    - "Why is the output of our program different from what we'd expect? What might we need to correct in our code?"
        - If everyone seems unsure what the problem is (the order of the if statements), it would be a good idea to do a quick code trace of the program
5.
    - Run the program and show that it works correctly after applying the changes (
    - "So what you all might've noticed is that while working on this we encountered three different types of errors. Let's go over in more detail what those three types of errors are." AKA segue into the slides

Activity 1: Code Bugging (7 - 10 minutes)

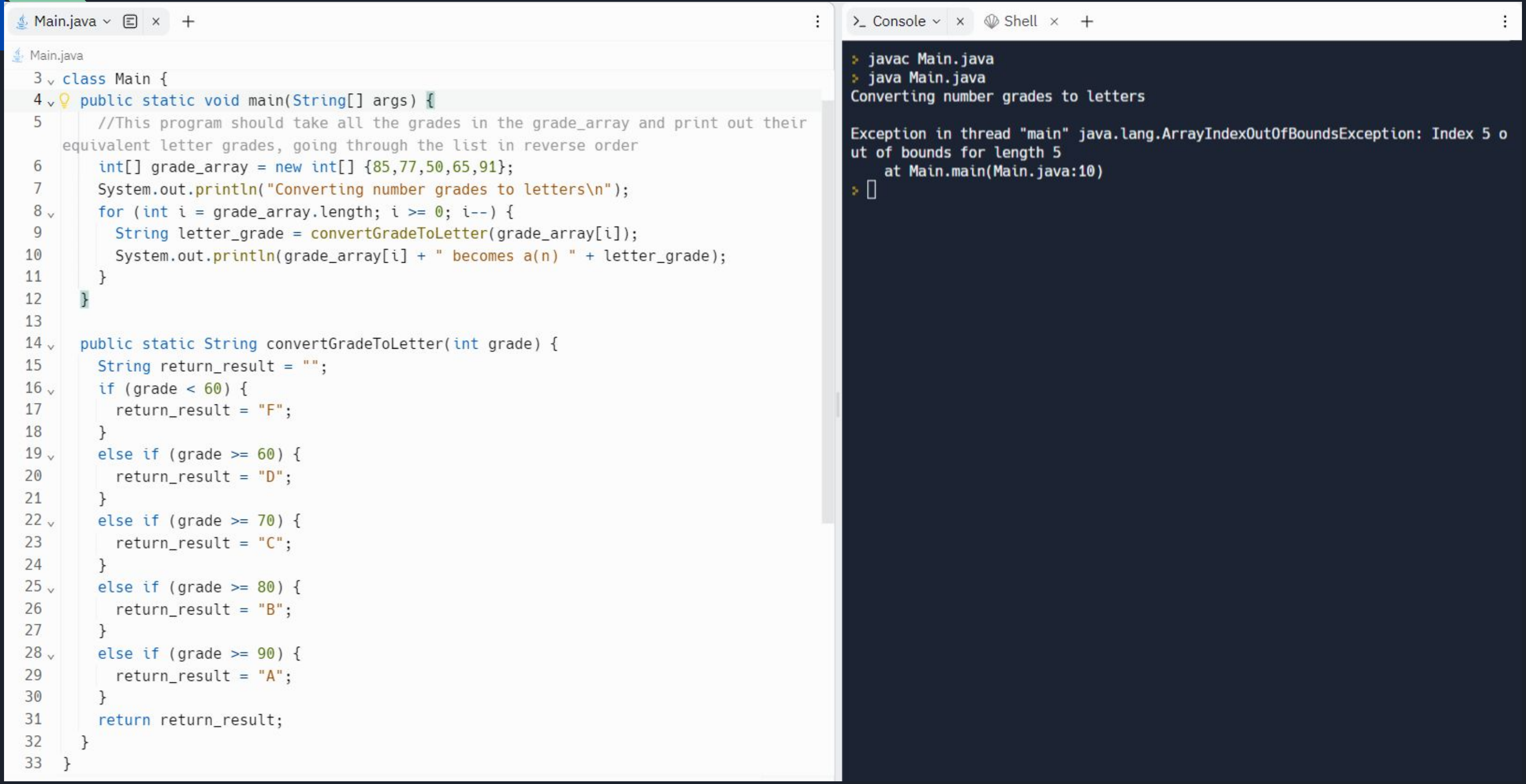
Present students with code (provided in materials as CurvedAverage) and ask them what potential errors we could introduce into this code. Make sure they also include what type of error they think this will cause and/or what they think the error message might look like (if any). Make sure to then run the newly bugged code live to see what error actually occurs

- If need be, ask for specific types of errors if students are focusing on one type, i.e. “Can anyone think of any runtime errors we could introduce into this code”.
- If you trust your students, you could also have students come up to the front of the class and add the errors themselves.
    - With this variation, you as the teacher could close your eyes and try to demonstrate how you might go about debugging such a program
    - Alternatively, you could have the rest of the class close their eyes and see if they can debug the newly introduced error.

Activity 2: Code Debugging (8 - 12 minutes)

Students will be presented with code (provided in materials as ReverseString) that is already riddled with errors (and will be told how many errors there are to look for). Students will take some time to look through and even run the code on their own and try to find and fix the errors themselves before reconvening as a class to go over what everyone found.

Do Now Image 1



Main.java

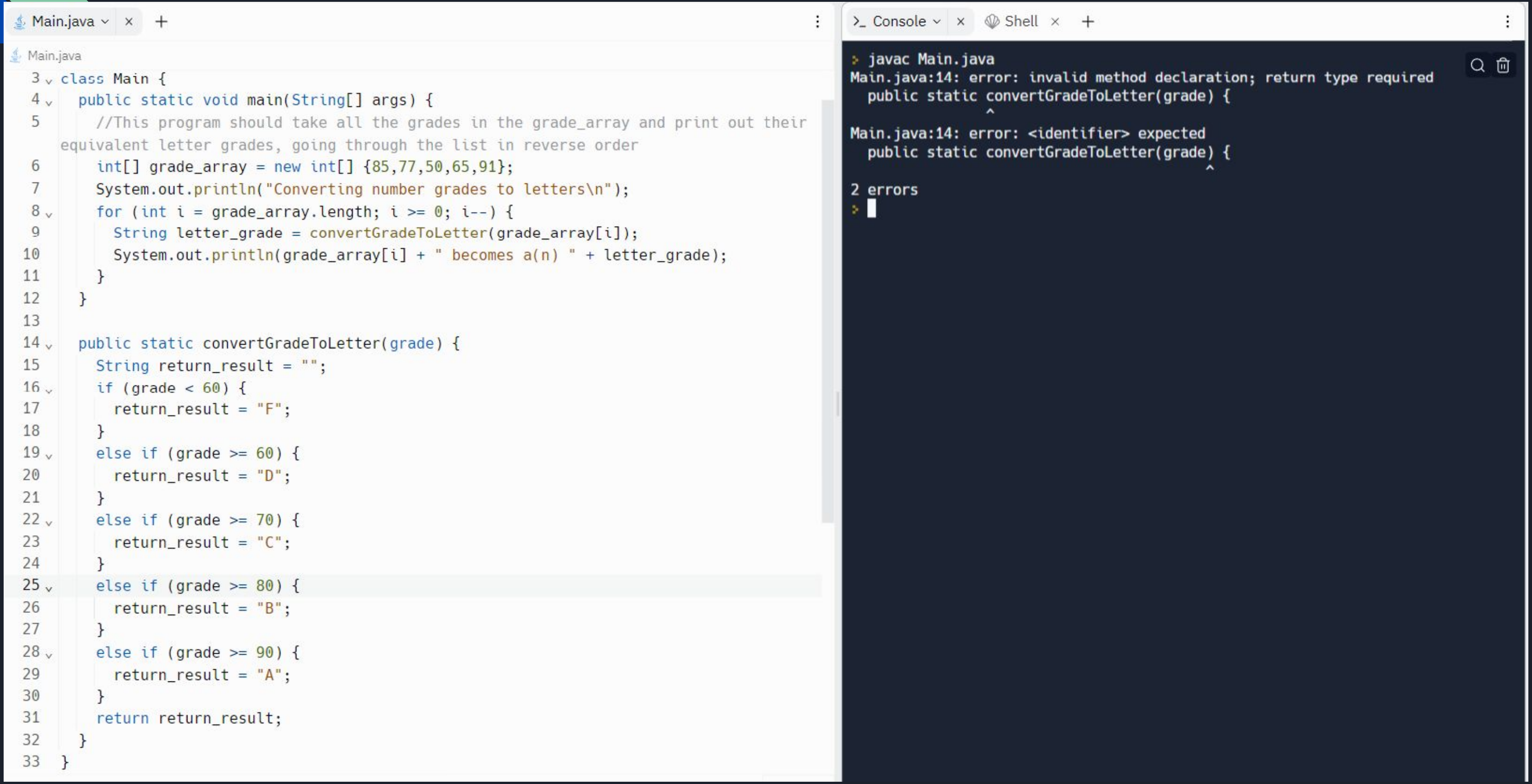
```java
class Main {
  public static void main(String[] args) {
    //This program should take all the grades in the grade_array and print out their equivalent letter grades, going through the list in reverse order
    int[] grade_array = new int[] {85,77,50,65,91};
    System.out.println("Converting number grades to letters\n");
    for (int i = grade_array.length; i >= 0; i--) {
      String letter_grade = convertGradeToLetter(grade_array[i]);
      System.out.println(grade_array[i] + " becomes a(n) " + letter_grade);
    }
  }

  public static String convertGradeToLetter(int grade) {
    String return_result = "";
    if (grade < 60) {
      return_result = "F";
    }
    else if (grade >= 60) {
      return_result = "D";
    }
    else if (grade >= 70) {
      return_result = "C";
    }
    else if (grade >= 80) {
      return_result = "B";
    }
    else if (grade >= 90) {
      return_result = "A";
    }
    return return_result;
  }
}
```

```
> javac Main.java
> java Main.java
Converting number grades to letters

Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: Index 5 o
ut of bounds for length 5
    at Main.main(Main.java:10)
>
```

Do Now Image 2



Main.java

```java
class Main {
  public static void main(String[] args) {
    //This program should take all the grades in the grade_array and print out their equivalent letter grades, going through the list in reverse order
    int[] grade_array = new int[] {85,77,50,65,91};
    System.out.println("Converting number grades to letters\n");
    for (int i = grade_array.length; i >= 0; i--) {
      String letter_grade = convertGradeToLetter(grade_array[i]);
      System.out.println(grade_array[i] + " becomes a(n) " + letter_grade);
    }
  }

  public static convertGradeToLetter(grade) {
    String return_result = "";
    if (grade < 60) {
      return_result = "F";
    }
    else if (grade >= 60) {
      return_result = "D";
    }
    else if (grade >= 70) {
      return_result = "C";
    }
    else if (grade >= 80) {
      return_result = "B";
    }
    else if (grade >= 90) {
      return_result = "A";
    }
    return return_result;
  }
}
```

```
> javac Main.java
Main.java:14: error: invalid method declaration; return type required
  public static convertGradeToLetter(grade) {
                ^
Main.java:14: error: <identifier> expected
  public static convertGradeToLetter(grade) {
                                          ^
2 errors
>
```

Three Types of Errors

- Syntax Errors
    - These are errors that tend to show up and prevent a program from compiling or run in the first place
- Runtime Errors
    - These are errors that tend to show up while a program is running, usually causing it to crash
- Logical Errors
    - These are errors that neither stop a program from compiling nor from crashing, but lead to different outcomes than otherwise expected.

```
import java.util.*;

class Main {

  public static void main(String[] args) {

    //This program should take all the grades in the
grade_array and print out their equivalent letter
grades, going through the list in reverse order

    int[] grade_array = new int[] {85,77,50,65,91};

    System.out.println("Converting number grades
to letters\n");

    for (int i = grade_array.length; i >= 0; i--) {

      String letter_grade =
convertGradeToLetter(grade_array[i]);

      System.out.println(grade_array[i] + " becomes
a(n) " + letter_grade);

    }

  }

  public static convertGradeToLetter(grade) {

    String return_result = "";

    if (grade < 60) {

      return_result = "F";

    }

    else if (grade >= 60) {

      return_result = "D";

    }

    else if (grade >= 70) {

      return_result = "C";

    }

    else if (grade >= 80) {

      return_result = "B";

    }

    else if (grade >= 90) {

      return_result = "A";

    }

    return return_result;

  }

}
```

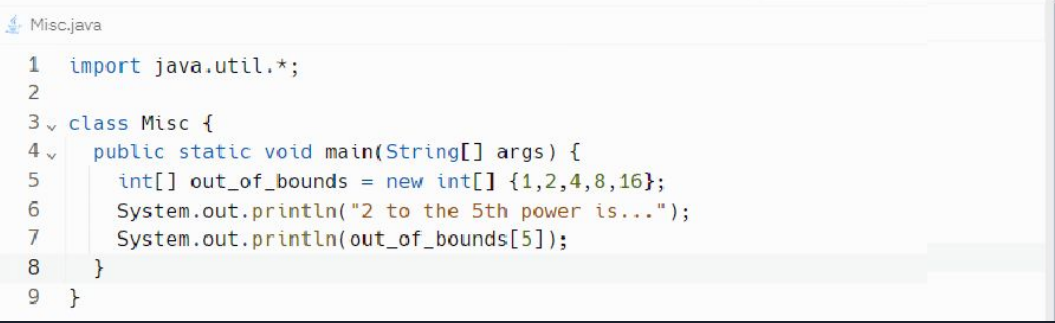
Syntax Errors

- Syntax errors, like the name implies, are errors that that arise from mistakes in following a programming language's syntactic structure.
- Syntax errors are generally detected at compile-time and prevent a program from executing its code.
- Syntax errors often occur due to spelling mistakes and/or inconsistencies (including capitalization) as well as missing, misplaced, and/or extraneous characters such as parentheses (), curly brackets {}, or quotes "".
- IDEs such as Repl.it can be very helpful in catching and correcting syntax errors like those above, but it can't catch everything.
- Other syntax errors are be dependent on the programming language, and what might be a syntax error in one language might be another type of error in another language or potentially not cause an error at all.
- In Java particularly, syntax errors can arise if one is not careful about data types.

Runtime Errors

- Even if no errors for a program are detected during compile-time, that doesn’t guarantee that a program is error free.
- Like the name implies, runtime errors are errors that occur while a program is running, generally causing the program to stop and preventing the execution of code after the source of the error.
- While runtime errors can occur for a number of reasons, the slides that follow will showcase some of the most common causes of runtime errors

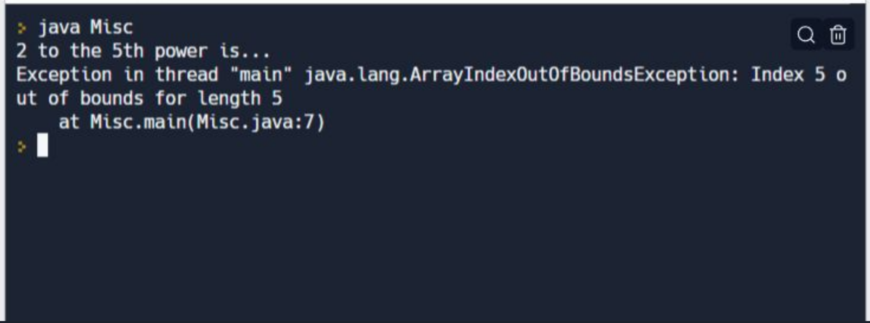
Runtime Errors (cont.)



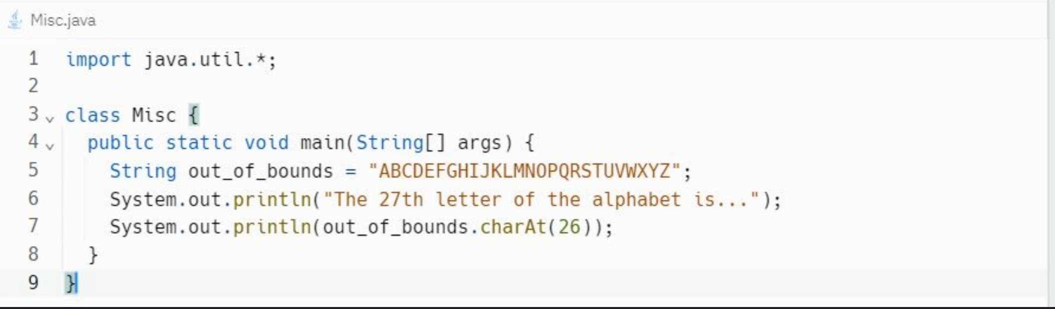
Misc.java

```java
import java.util.*;

class Misc {
  public static void main(String[] args) {
    int[] out_of_bounds = new int[] {1,2,4,8,16};
    System.out.println("2 to the 5th power is...");
    System.out.println(out_of_bounds[5]);
  }
}
```



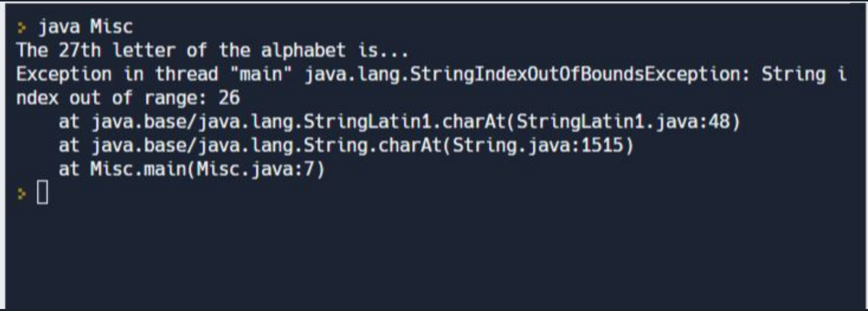
```
> java Misc
2 to the 5th power is...
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: Index 5 o
ut of bounds for length 5
	at Misc.main(Misc.java:7)
>
```



Misc.java

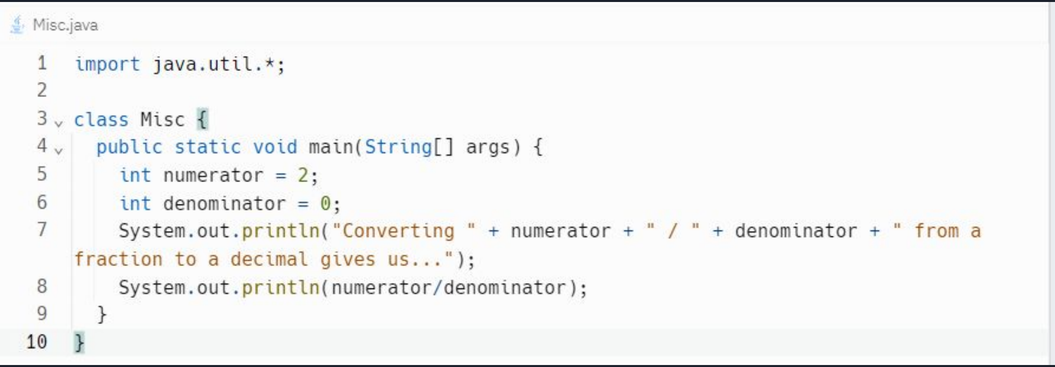
```java
import java.util.*;

class Misc {
  public static void main(String[] args) {
    String out_of_bounds = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
    System.out.println("The 27th letter of the alphabet is...");
    System.out.println(out_of_bounds.charAt(26));
  }
}
```



```
> java Misc
The 27th letter of the alphabet is...
Exception in thread "main" java.lang.StringIndexOutOfBoundsException: String i
ndex out of range: 26
	at java.base/java.lang.StringLatin1.charAt(StringLatin1.java:48)
	at java.base/java.lang.String.charAt(String.java:1515)
	at Misc.main(Misc.java:7)
>
```

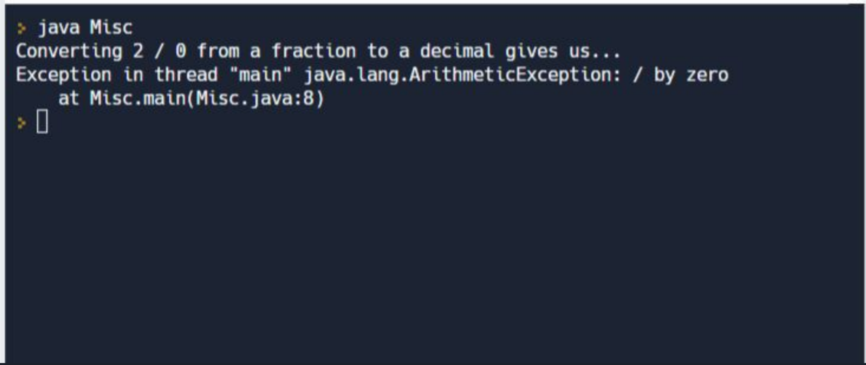
Runtime Errors (cont.)



Misc.java

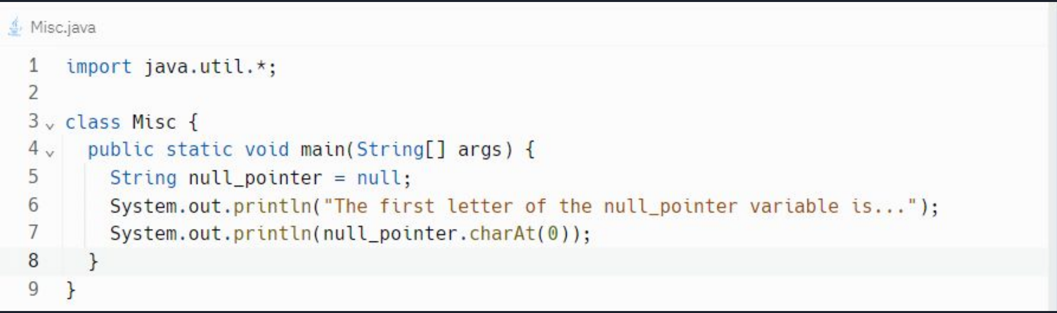
```java
import java.util.*;

class Misc {
  public static void main(String[] args) {
    int numerator = 2;
    int denominator = 0;
    System.out.println("Converting " + numerator + " / " + denominator + " from a fraction to a decimal gives us...");
    System.out.println(numerator/denominator);
  }
}
```



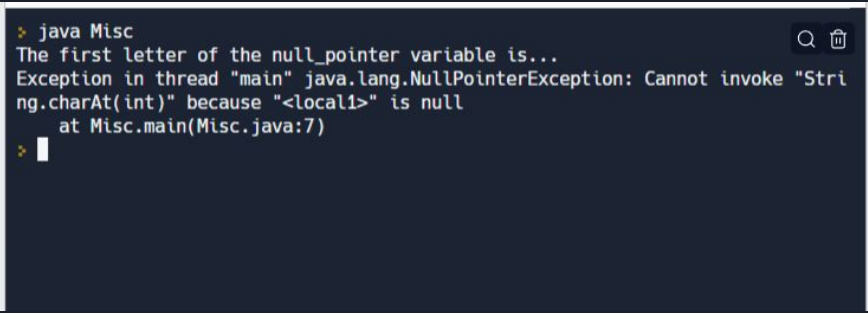
```
> java Misc
Converting 2 / 0 from a fraction to a decimal gives us...
Exception in thread "main" java.lang.ArithmeticException: / by zero
    at Misc.main(Misc.java:8)
> 
```

Runtime Errors (cont.)


Misc.java

```java
import java.util.*;

class Misc {
  public static void main(String[] args) {
    String null_pointer = null;
    System.out.println("The first letter of the null_pointer variable is...");
    System.out.println(null_pointer.charAt(0));
  }
}
```

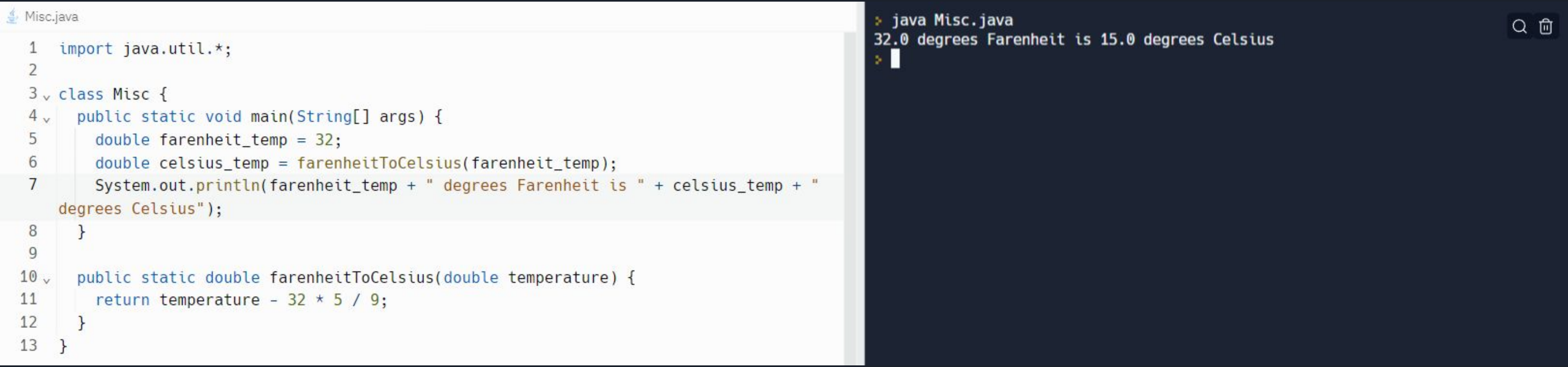


```
> java Misc
The first letter of the null_pointer variable is...
Exception in thread "main" java.lang.NullPointerException: Cannot invoke "Stri
ng.charAt(int)" because "<local1>" is null
    at Misc.main(Misc.java:7)
> 
```

Logic Errors

- Remember that computers will do what we tell them to do which is not always necessarily what we want them to do.
- Logic errors aren't errors in the same way that syntax errors and runtime errors are. Logic errors refer to instances where a program behaves differently than expected.
- Logic errors can vary greatly from using the wrong variable by accident to having an incorrect algorithm altogether.
- The following slides contain some examples of logic errors although it is by no means an exhaustive list.

Logic Errors (cont.)



Misc.java

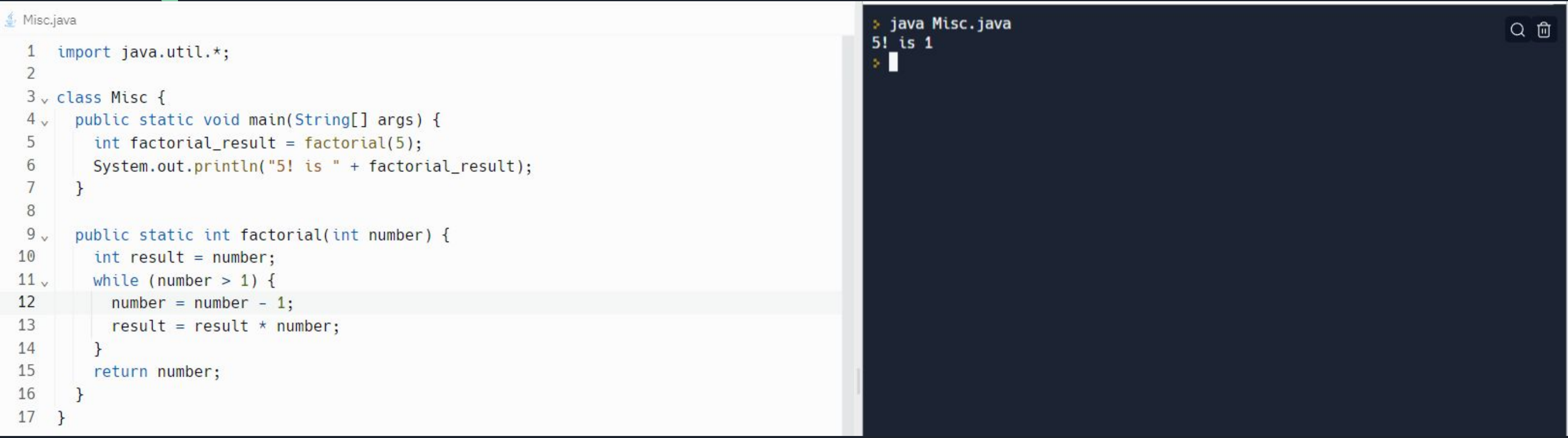
```java
import java.util.*;

class Misc {
  public static void main(String[] args) {
    double farenheit_temp = 32;
    double celsius_temp = farenheitToCelsius(farenheit_temp);
    System.out.println(farenheit_temp + " degrees Farenheit is " + celsius_temp + " degrees Celsius");
  }

  public static double farenheitToCelsius(double temperature) {
    return temperature - 32 * 5 / 9;
  }
}
```

```
> java Misc.java
32.0 degrees Farenheit is 15.0 degrees Celsius
>
```

Logic Errors (cont.)


Misc.java

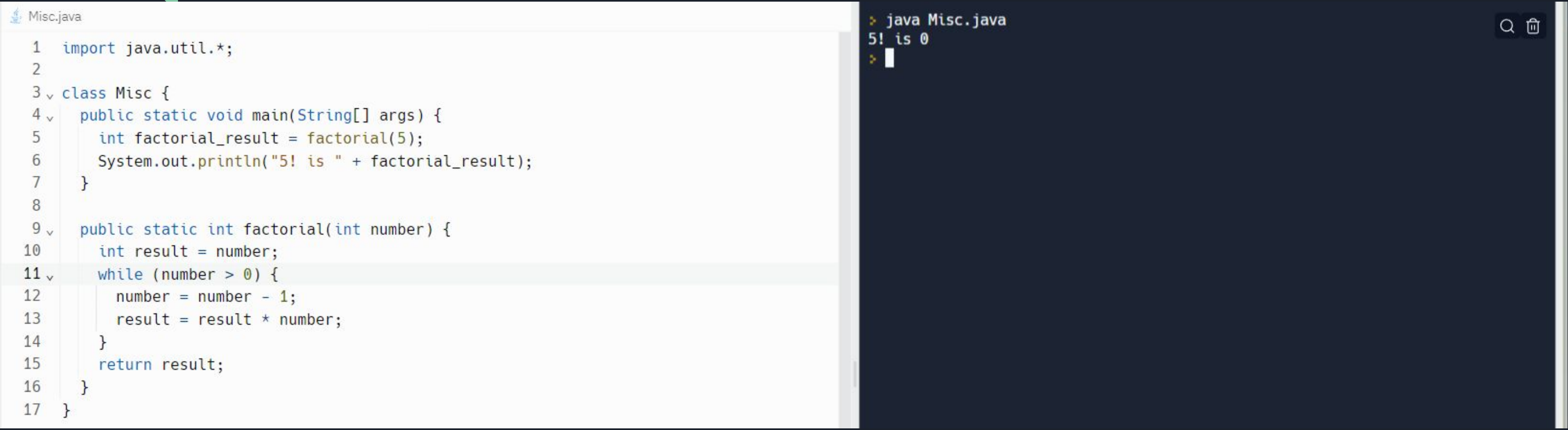
```java
import java.util.*;

class Misc {
  public static void main(String[] args) {
    int factorial_result = factorial(5);
    System.out.println("5! is " + factorial_result);
  }

  public static int factorial(int number) {
    int result = number;
    while (number > 1) {
      number = number - 1;
      result = result * number;
    }
    return number;
  }
}
```

```
> java Misc.java
5! is 1
> 
```

Logic Errors (cont.)



Misc.java

```java
import java.util.*;

class Misc {
  public static void main(String[] args) {
    int factorial_result = factorial(5);
    System.out.println("5! is " + factorial_result);
  }

  public static int factorial(int number) {
    int result = number;
    while (number > 0) {
      number = number - 1;
      result = result * number;
    }
    return result;
  }
}
```

```
> java Misc.java
5! is 0
>
```

The following slides have the code materials needed for the lesson. They are in very small font but they should be copy and pastable nonetheless. In Order they are

1. ReverseArray
2. CurvedAverage
3. CurvedAverageStudent

```
import java.util.*;

class ReverseArray {

 public static void main(String[] args) {

  //This program should an array of Strings and print them out as a sentence, but in reverse order.

  String[] sentence_array = new String[] {"Us","Among","Imposter","1","Is","There"};

  int front_index = 0;

  int back_index = sentence_array.length - 1;

  while (front_index < back_index) {

   swap(sentence_array,front_index,back_index);

   front_index = front_index + 1;

   back_index = back_index - 1;

  }



  System.out.println("Printing the array in reverse as a sentence");

  for (int i = 0; i < sentence_array.length; i++) {

   System.out.print(sentence_array[i] + " ");

  }

 }



 public static void swap(String[] string_array, int index1, int index2) {

  String temp = string_array[index1];

  string_array[index1] = string_array[index2];

  string_array[index2] = temp;

 }
```

```
import java.util.*;

class CurvedAverage {

 public static void main(String[] args) {

  //This program first makes an array of 5 random numbers between 50 and 100

  Random rand = new Random();

  int[] grade_array = new int[5];

  for (int i = 0; i < 5; i++ ) {

   grade_array[i] = 50 + rand.nextInt(50);

  }

  int sum = 0;

  Scanner user_input = new Scanner(System.in);

  //Once the array of grades is made, the program goes through the array again, asking for each grade how many points should be added. The program should then calculate the average of all the grades, after extra credit has been applied

  for (int i = 0; i < grade_array.length; i++) {

   System.out.println("The current grade is " + grade_array[i] + ". How many points do you wish to add? ");

   int extra_credit = user_input.nextInt();

   grade_array[i] = grade_array[i] + extra_credit;

   sum = sum + grade_array[i];

  }

  double average = (sum / (double) grade_array.length);

  System.out.println("The new average is " + average);

 }
```

```
import java.util.*;
class CurvedAverageStudent {
  public static void main(String[] args) {
    //This program first makes an array of 5 random numbers between 50 and 100
    Random rand = new Random;
    grade_array = new int[5];
    for (int i = 1; i < 5; i++ ) {
      grade_array[i] = 50 + rand.nextInt(50);
    }
    int sum;
    Scanner user_input = new Scanner(System.in);
    //Once the array of grades is made, the program goes through the array again, asking for each grade how many points should be added. The program should then calculate the average of all the grades, after extra credit has been applied
    for (int i = 0; i <= grade_array.length; i++) {
      System.out.println("The current grade is " + grade_array[i] + ". How many points do you wish to add? ");
      int extra_credit = user_input.nextInt();
      sum = sum + grade_array[i];
      grade_array[i] = grade_array[i + extra_credit];
    }
    int average = (sum / (double) grade_array.length);
    System.out.println("The new average is " + average);
  }
```