Lecture Feb 5 - Strings, Characters, and Errors

Bentley James Oakes

February 5, 2018

Assignment 1

- The assignments are being marked now
- They'll be returned next week, before the next due date
- You'll receive feedback about your code
- Solutions posted soon

Assignment 2

- Posted now
- Due **Thursday, February 15th**
- You've seen all the material

Catch-up

- Last class and this one are mainly focused on examples
  - We will talk about `Strings` and `characters` today
- These lectures are to provide a chance to catch up
- I'll be assuming you understand the past material

This Lecture

1 Example: Collatz Conjecture

2 Iterating Through Strings

3 Example: Reversing a String

4 Testing Characters

5 Examples: Converting Binary-to-Decimal

6 Errors

7 Exceptions

Section 1

Example: Collatz Conjecture

Collatz Conjecture

- Let's look at an math problem which is unsolved
- It's called the *Collatz Conjecture*
    - Conjecture is another word for theory

"this is an extraordinarily difficult problem, completely out of reach of present day mathematics."

Don't worry, it's easy to explain!

Collatz Conjecture

The conjecture states:

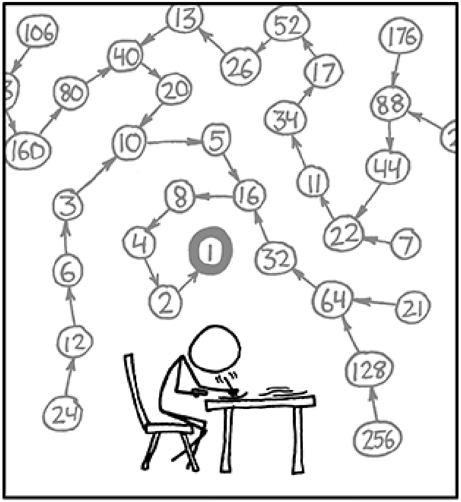
- Start with a number n
- If n is even, divide n by 2
- If n is odd, multiply n by 3 and add 1
- **No matter what n you start with, eventually a 1 is produced**

Examples:

- Starting with $n = 12$ gives the sequence 12, 6, 3, 10, 5, 16, 8, 4, 2, 1
- $n = 19$ takes a longer time to end: 19, 58, 29, 88, 44, 22, 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1.

So is there any sequence that doesn't end in a 1?

Collatz Conjecture XKCD



"The Collatz Conjecture states that if you pick a number,

and if it's even divide it by two and if it's odd, multiply it by three and add one,

and you repeat this procedure long enough,

eventually your friends will stop calling to see if you want to hang out."

Collatz Program

- We won't solve this hard problem
- But let's write a method that generates the Collatz sequence for a particular number

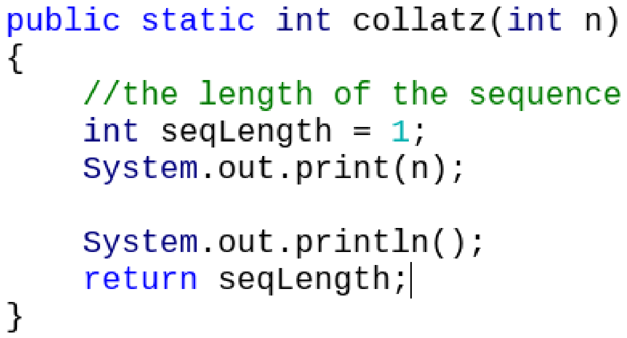
- **Method Name:** `collatz`
- **Parameter:** `int n`
- **Return Type:** `int`
- **Description:** Takes in a `int` *n*, prints out the numbers in the Collatz sequence, and returns an `int` which is the length of that sequence
- **Example:** `collatz(12)` prints out 12, 6, 3, 10, 5, 16, 8, 4, 2, 1, and returns 10

Planning Out Our Method

- So we start with our number *n*
- Until *n* becomes one
    - If *n* is even
        - Divide it by two
    - If *n* is odd
        - Multiply it by three and add one
    - Print each step in this sequence
    - Count how many items are in this sequence
- Return the count of items

Collatz Conjecture

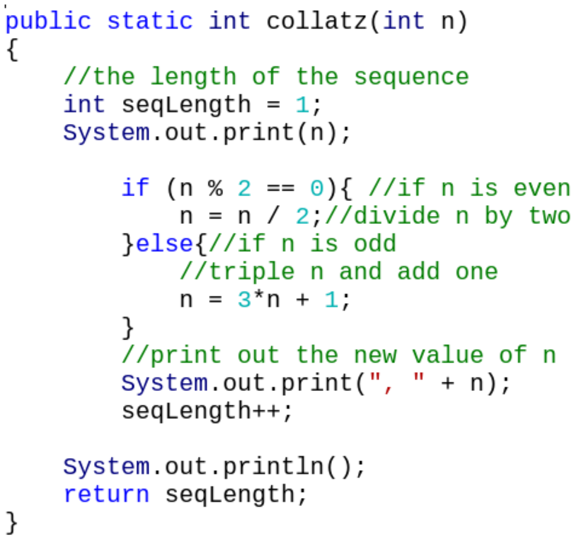
- Again start with just enough to compile



```
public static int collatz(int n)
{
    //the length of the sequence
    int seqLength = 1;
    System.out.print(n);

    System.out.println();
    return seqLength;
}
```

Collatz Conjecture

- Hard to divide this up into smaller steps
- This produces the next step in the sequence



```java
public static int collatz(int n)
{
    //the length of the sequence
    int seqLength = 1;
    System.out.print(n);

        if (n % 2 == 0){ //if n is even
            n = n / 2;//divide n by two
        }else{//if n is odd
            //triple n and add one
            n = 3*n + 1;
        }
        //print out the new value of n
        System.out.print(", " + n);
        seqLength++;

    System.out.println();
    return seqLength;
}
```

Collatz Conjecture

- Add the while loop to test if *n* is one

```
public static int collatz(int n)
{
    //the length of the sequence
    int seqLength = 1;
    System.out.print(n);

    while (n != 1){ //while n is not 1
        if (n % 2 == 0){ //if n is even
            n = n / 2;//divide n by two
        }else{//if n is odd
            //triple n and add one
            n = 3*n + 1;
        }
        //print out the new value of n
        System.out.print(", " + n);
        seqLength++;
    }

    System.out.println();
    return seqLength;
}
```

Collatz Conjecture

- Let's add another method
- Now that we can get the length of the Collatz sequence for a number,
- **Which number between 1 and 100 has the longest sequence?**

- Method Name: `maxLength`
- Parameter: `int topNum`
- Return Type: `int`
- Description: Given the parameter `int topNum`, this method returns the `int` number between 1 and `topNum` which has the longest Collatz sequence

Collatz Conjecture

```
public static int maxLength(int topNum)
{
    //save the longest sequence length
    //and the number who's sequence it is
    int numberWithLongestSequence = 1;
    int longestSequenceLength = 1;

    //loop through from 1 to topNum
    for (int i=1; i <= topNum; i++)
    {
        //get the length
        int length = collatz(i);

        //if the length is bigger than
        //the stored length,
        //update the longest length
        //and the longest number
        if (length > longestSequenceLength){
            longestSequenceLength = length;
            numberWithLongestSequence = i;
        }
    }
    return numberWithLongestSequence;
}
```

Collatz Conjecture

- The main method to call the other two methods

```
public static void main(String[] args)
{
    int longest = maxLength(100);
    System.out.println("The num with longest sequence is: " + longest);
    System.out.println("The length is: " + collatz(longest));
}
```

Section 2

Iterating Through Strings

Motivation

- To really show off the power of *for loops*
- Let's try iterating through Strings
- Things we can do:
  - Count the number of vowels
  - Reverse the String
  - See if the String is a palindrome
    - Same forward and backward

String Length

- Let's first print out the length of a string

```
String s = "Hello World!";
int strLength = s.length();
System.out.println("Length: " + strLength);
//Length: 12
```

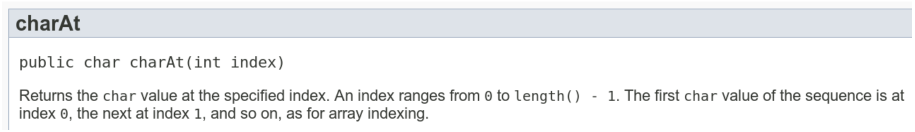
- Note that we are calling a method on a variable
- You saw this before, with the equals method
  - `s.equals(“hello”)`

String Length Examples

```
String one = "Good ";
String two = "Evening";
String together = one + two;

int togetherLength = together.length();
int anotherWay = one.length() + two.length();

System.out.println("Together: " + togetherLength);
System.out.println("Another Way: " + anotherWay);

//prints Together: 12
//prints Another Way: 12
```

charAt Method

- To iterate through a String properly, we also need to get characters out of the String
- This is the `charAt` method
- https://docs.oracle.com/javase/9/docs/api/java/lang/String.html



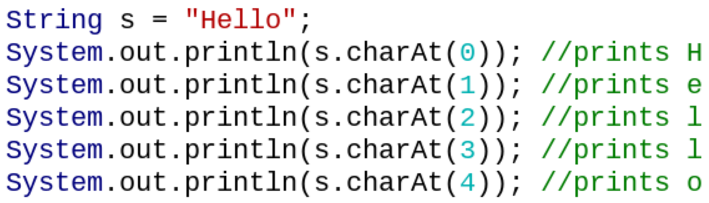
charAt

```
public char charAt(int index)
```

Returns the `char` value at the specified index. An index ranges from `0` to `length() - 1`. The first `char` value of the sequence is at index `0`, the next at index `1`, and so on, as for array indexing.

charAt Examples

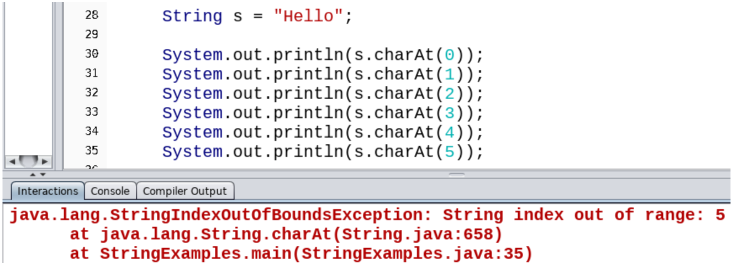
- The `charAt method` takes an int, which is the **index** or position along the String



```
String s = "Hello";
System.out.println(s.charAt(0)); //prints H
System.out.println(s.charAt(1)); //prints e
System.out.println(s.charAt(2)); //prints l
System.out.println(s.charAt(3)); //prints l
System.out.println(s.charAt(4)); //prints o
```

- What's at `s.charAt(5)`?

Index Error

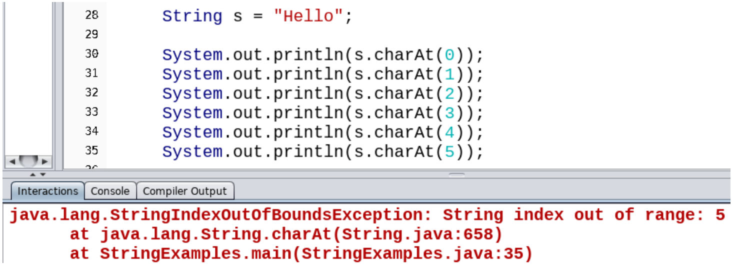

```
        String s = "Hello";

        System.out.println(s.charAt(0));
        System.out.println(s.charAt(1));
        System.out.println(s.charAt(2));
        System.out.println(s.charAt(3));
        System.out.println(s.charAt(4));
        System.out.println(s.charAt(5));
```

```
java.lang.StringIndexOutOfBoundsException: String index out of range: 5
	at java.lang.String.charAt(String.java:658)
	at StringExamples.main(StringExamples.java:35)
```

- Very common error
- Known as an 'off-by-one error'

There are only two hard things in computer science: cache invalidation, naming things, and off-by-one errors.

length() - 1



```
String s = "Hello";

System.out.println(s.charAt(0));
System.out.println(s.charAt(1));
System.out.println(s.charAt(2));
System.out.println(s.charAt(3));
System.out.println(s.charAt(4));
System.out.println(s.charAt(5));
```

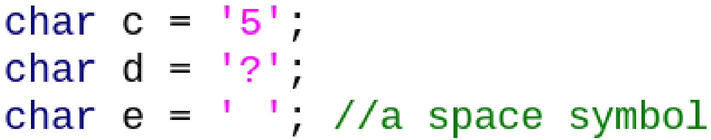
```
java.lang.StringIndexOutOfBoundsException: String index out of range: 5
      at java.lang.String.charAt(String.java:658)
      at StringExamples.main(StringExamples.java:35)
```

- Note that the `String` has a length of 5
- `char x = s.charAt(s.length());` is an indexing error
- `char c = s.charAt(s.length() - 1);` is the last character in the `String`

About Chars

- The return type of the `charAt` method is another variable type:
- **char** - for characters
- The char type stores one letter or symbol

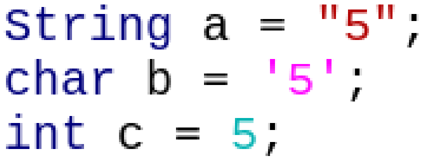
For example:



```
char c = '5';
char d = '?';
char e = ' '; //a space symbol
```

Strings vs chars

- Make sure that you understand the difference between a char and a String
- A `char` is a symbol, a `String` is a collection of symbols



```
String a = "5";
char b = '5';
int c = 5;
```

- These are three different types
- The values are not equal to each other!
- Note that `chars` are made with one quotation mark, and `Strings` have two

Objects

- You'll notice that `Strings` seem to be more complicated than ints or doubles
- `Strings` are made up of chars, and you can call methods on `Strings` (such as .equals())
- `Strings` are actually *Objects*, which will we talk about later in the course
  - Bottom-line: Objects can be very complex, and have methods associated with them

Iterating Through Strings

- Now that we know the beginning and end of a String,
- We can iterate through in a *for loop*
- to print every character in the String

```
String s = "this is a string.";
for (int i=0; i < s.length(); i++)
{
    //print out the character
    System.out.print("<" + s.charAt(i) + ">");
}
System.out.println();
```

```
<t><h><i><s>< ><i><s>< ><a>< ><s><t><r><i><n><g><.>
```

Iterating Backwards

- We can iterate backwards through Strings too
- Pay attention to the creation/condition/modification components of the *for loop*!

```
String s = "this is a string.";
for (int i=s.length()-1; i >= 0; i--)
{
    //print out the character
    System.out.print("<" + s.charAt(i) + ">");
}
System.out.println();
```

```
<.><g><n><i><r><t><s>< ><a>< ><s><i>< ><s><i><h><t>
```

Section 3

Example: Reversing a String

Reversing a String

- Let's write a method to reverse a String

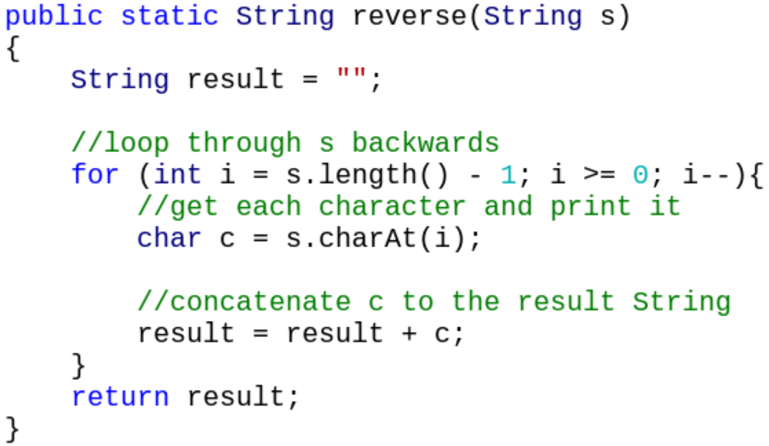
- **Method Name:** `reverse`
- **Parameter:** `String s`
- **Return Type:** `String`
- **Description:** Takes in a `String`, and returns a `String` which has all the characters in reverse order
- **Example:** `reverse(‘‘hello!’’)` returns “!olleh”

Planning Out Our Method

- The input is our `String s`
- We'll create an empty `String` called `result`
- Then we'll loop through `s` from end to beginning
    - Get each character in `s`
    - Add the character to `result`
- Return `result`

Reverse String

- Loop through the `String` backwards
- Add each character to `result` using concatenation



```java
public static String reverse(String s)
{
    String result = "";

    //loop through s backwards
    for (int i = s.length() - 1; i >= 0; i--){
        //get each character and print it
        char c = s.charAt(i);

        //concatenate c to the result String
        result = result + c;
    }
    return result;
}
```

Section 4

Testing Characters

Testing Characters

- Now that we have characters, we can start testing them
- Let's write a method to see if a String contains either the 'a' or 'A' characters

ContainsA

```
//returns true if the String contains
//upper or lowercase 'a'
public static boolean containsA(String s)
{
    //loop through the String
    for (int i=0; i < s.length(); i++){

        //put each character in c
        char c = s.charAt(i);

        //if this character is c
        //return true
        if (c == 'a' || c == 'A'){
            return true;
        }
    }

    //if we made it through the loop, return False
    return false;
}
```

char Comparisons

- We can also compare chars
- Let's write a method to count how many letters
- and how many numbers appear in a String

Note

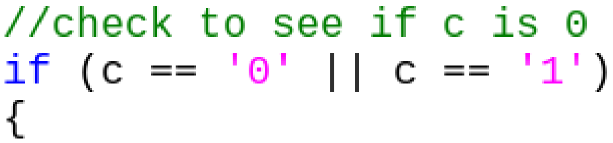
```
if ('a' <= c && c <= 'z'){
    numLetters ++;
}
```

- We can compare characters using $<$ and $>$
  - This test sees if c is a lower-case letter
- Note that chars literals have a single quotation mark ', not "

CountString

```
public static void countString(String s)
{
    int numLetters = 0;
    int numNumbers = 0;

    //loop through the String
    for (int i=0; i < s.length(); i++){
        //put each character in c
        char c = s.charAt(i);
        //if this character is a lowercase
        //letter increase the count
        if ('a' <= c && c <= 'z'){
            numLetters ++;
        }
        //if this character is a number
        //increase the count
        if ('0' <= c && c <= '9'){
            numNumbers ++;
        }
    }
    System.out.println("String: " + s);
    System.out.println("Num Letters: " + numLetters);
    System.out.println("Num Numbers: " + numNumbers);
}
```

Zero or One Check



```
//check to see if c is 0 or 1
if (c == '0' || c == '1')
{
```

- Let's use this in an example of converting binary to decimal

Section 5

Examples: Converting Binary-to-Decimal

Binary-to-Decimal Converter

- These slides will show the progression of building a binary-to-decimal converter method

- Method Name: `convertBinToDec`
- Parameter: `String binString`
- Return Type: `int`
- Description: Takes in a `String` which contains a binary number, and returns an `int` which contains the decimal value of that binary number
- Example: `convertBinToDec(‘‘1010’’) returns 10`

Binary-to-Decimal

- Input: A binary number
- Instructions:
  - Write the powers of 2 below each digit
  - Starting with 1 to the far right
  - Add up the powers of 2 if a 1 appears in the binary number
  - This sum is the answer
- Output: The number in decimal

Planning Out Our Method

- So we need to iterate through the binary number
- For each character in the number:
    - Figure out the power-of-two for that location
    - If the character is a one:
        - Add the power of two to a result variable
- This result variable has our answer

Binary-to-Decimal Converter

- To start, let's write just enough of the method so that it compiles

```
public static int convertBinToDec(String binaryString)
{
    int result = 0;

    return result;
}
```

Binary-to-Decimal Converter

- Let's then iterate from the end of the String to the beginning,
- And print out each character in the String

```java
public static int convertBinToDec(String binaryString)
{
    int result = 0;

    //iterate from the end of the String to the beginning
    for (int i = binaryString.length() -1; i >= 0 ; i--){

        //get each character
        char c = binaryString.charAt(i);

        //print it
        System.out.println(c);
    }

    return result;
}
```

Binary-to-Decimal Converter

```
public static int convertBinToDec(String binaryString)
{
    int result = 0;

    int powerOfTwo = 1;

    //iterate from the end of the String to the beginning
    for (int i = binaryString.length() -1; i >= 0 ; i--){

        //get each character
        char c = binaryString.charAt(i);

        //print it and the power of Two
        System.out.println(c + " " + powerOfTwo);

        //increase the powerOfTwo variable
        powerOfTwo = powerOfTwo * 2;
    }

    return result;
}
```

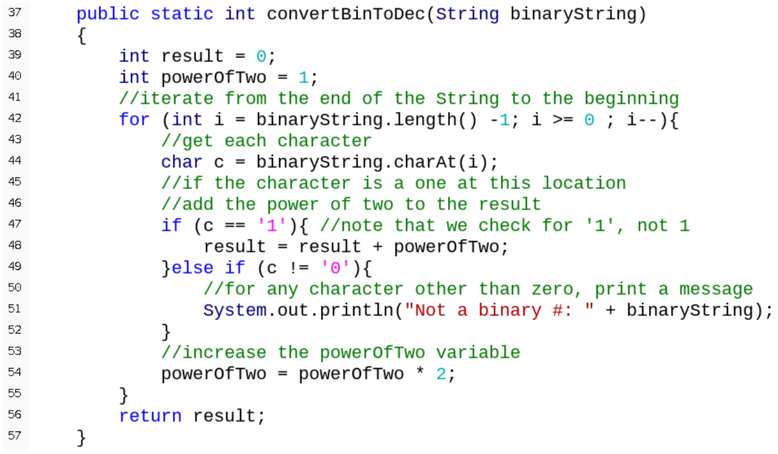
```
1 1
0 2
1 4
0 8
1 16
0 32
1 64
0 128
```

- We increase the power of two for every position we move
- We start at one, and double it

Binary-to-Decimal Converter

```
public static int convertBinToDec(String binaryString)
{
    int result = 0;

    int powerOfTwo = 1;

    //iterate from the end of the String to the beginning
    for (int i = binaryString.length() -1; i >= 0 ; i--){

        //get each character
        char c = binaryString.charAt(i);

        //if the character is a one at this location
        //add the power of two to the result
        if (c == '1'){
            result = result + powerOfTwo;
        }

        //increase the powerOfTwo variable
        powerOfTwo = powerOfTwo * 2;
    }

    return result;
}
```

Binary-to-Decimal Converter



```java
    public static int convertBinToDec(String binaryString)
    {
        int result = 0;
        int powerOfTwo = 1;
        //iterate from the end of the String to the beginning
        for (int i = binaryString.length() -1; i >= 0 ; i--){
            //get each character
            char c = binaryString.charAt(i);
            //if the character is a one at this location
            //add the power of two to the result
            if (c == '1'){ //note that we check for '1', not 1
                result = result + powerOfTwo;
            }else if (c != '0'){
                //for any character other than zero, print a message
                System.out.println("Not a binary #: " + binaryString);
            }
            //increase the powerOfTwo variable
            powerOfTwo = powerOfTwo * 2;
        }
        return result;
    }
```

- If the character is not a '0' or '1', print a message to let the user know they have entered something invalid
- Later, we'll see how to crash the program in this case

Section 6

Errors

Four Types of Errors

1. Logic error
2. Style error
3. Compiler error
4. Run-time error

- Often comes up on tests...
- We'll show you code, and ask what kind of error (if any) will occur

Logic Error

- **Logic errors** occur when the program does something different than what the programmer expects
- Example:
  - Taking the average of two numbers
  - Should be `int average = (a + b) / 2;`
  - But accidentally wrote `int average = a + b / 2;`
- Division happens first
- Java has no idea there's an issue
  - Instructions are valid, but the result is 'wrong'

Debugging

- The way to find logic errors is to think about what the code is doing
- And to **debug your code**
- Two ways to debug your code

1. The print statement
    - Print variable values at specific points in the program
    - Are the values what they should be?
2. Use a debugger

Dr. Java Debugger

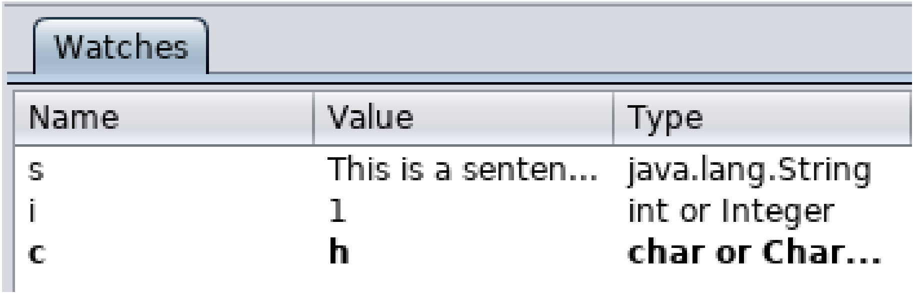
1. Go into *Debug Mode*. This is the first option in the Debugger drop-down menu.
2. Once in this mode, toggle a **breakpoint**. Click on a line of code and select the *Toggle Breakpoint on Current Line* option
   - A breakpoint is where code execution will stop in debug mode
   - Need at least one breakpoint to debug or the execution won't stop
3. Run your code from inside *Debug Mode*
4. While you are debugging, you can *step over*, *step into*, *resume*

Debugger

- **Step over:** Clicking this button will execute the current line of code and step to the next line in the current method
- **Step into:** If there is a method call on the current line, the debugger will step into the first line of the method
- **Resume:** Continue execution until the next breakpoint, or until the end of the program

Watches

- **Watches** let you keep trace of the values of variables
- Add the names of variables to the *Watches* list
- Very helpful if you think there's an *infinite loop* in your code
    - Infinite loop: where the loop never stops



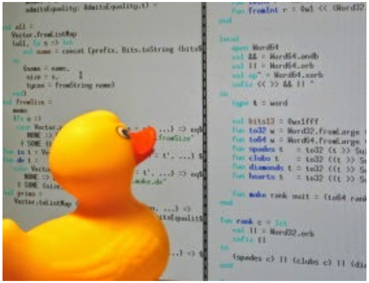
Watches

Name	Value	Type
s	This is a senten...	java.lang.String
i	1	int or Integer
c	**h**	**char or Char...**

Tips for Programming

- Always write code incrementally
  - Write some code, then test it
  - Keep adding small pieces and testing them
  - Try to write lots of methods
    - Every method should have one well-defined purpose

Rubber-duck Debugging



Explaining the problem to someone else is surprisingly useful.
`https://en.wikipedia.org/wiki/Rubber_duck_debugging`

Style Error

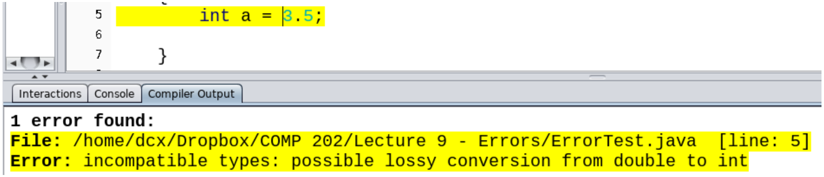
A **style error** is where the program is correct, but is hard to understand
Examples:

- Bad variable names
- Incorrect indentation
- Inconsistent braces
- Lack of comments
- Too many calculations on one line

- Java doesn't care, but it makes reading the program difficult

Compiler error

- **Compiler errors** happen during compilation
- The compiler examines your code and raises errors
- Mostly related to variable types, syntax errors, or typos


```
        int a = 3.5;

    }
```

```
1 error found:
File: /home/dcx/Dropbox/COMP 202/Lecture 9 - Errors/ErrorTest.java  [line: 5]
Error: incompatible types: possible lossy conversion from double to int
```

Dealing with Errors

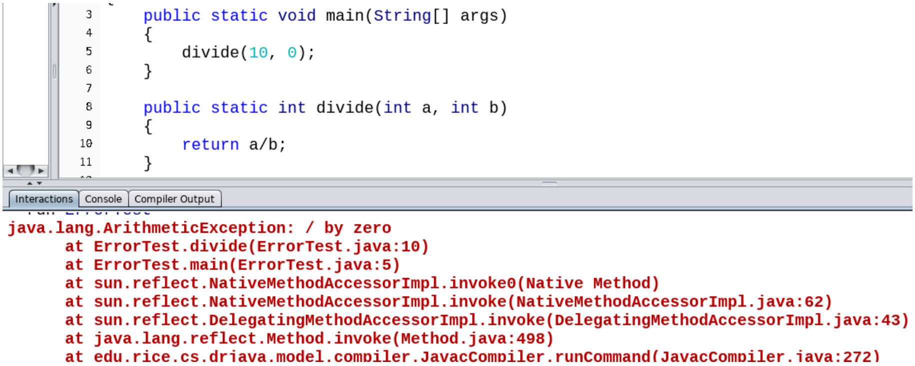
Tips for helping with compiler errors:

1. Always fix the errors from the top to the bottom
    - If a line has a problem, then the compiler gets confused
    - And might report following lines as having problems
2. Use the Internet to search for the error message

Run-time Error

- **Run-time errors** occur during the running of the program
- Java complains that something unexpected happened
- Most run-time errors depend on values of variables
- The compiler doesn't check the values for you

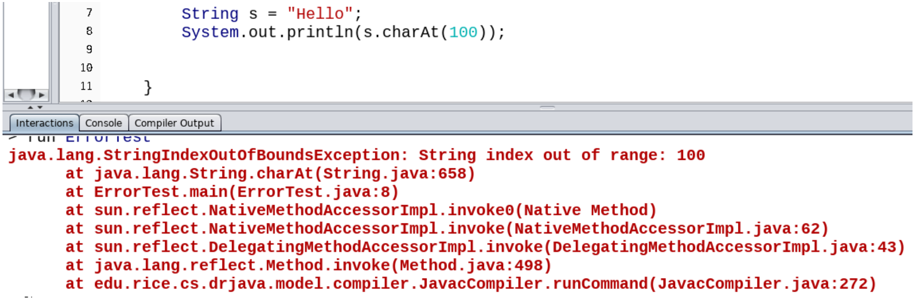
Run-time Error Example



```
    public static void main(String[] args)
    {
        divide(10, 0);
    }

    public static int divide(int a, int b)
    {
        return a/b;
    }
```

```
java.lang.ArithmeticException: / by zero
	at ErrorTest.divide(ErrorTest.java:10)
	at ErrorTest.main(ErrorTest.java:5)
	at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
	at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
	at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
	at java.lang.reflect.Method.invoke(Method.java:498)
	at edu.rice.cs.drjava.model.compiler.JavacCompiler.runCommand(JavacCompiler.java:272)
```

Run-time Error Example

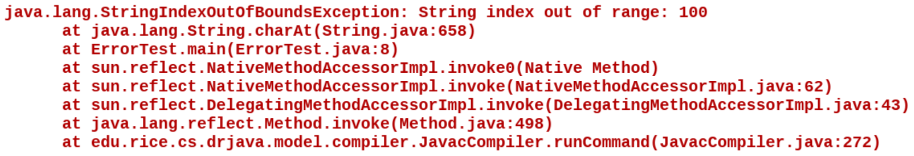


```
        String s = "Hello";
        System.out.println(s.charAt(100));


    }
```

Interactions | Console | Compiler Output

```
> run ErrorTest
java.lang.StringIndexOutOfBoundsException: String index out of range: 100
	at java.lang.String.charAt(String.java:658)
	at ErrorTest.main(ErrorTest.java:8)
	at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
	at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
	at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
	at java.lang.reflect.Method.invoke(Method.java:498)
	at edu.rice.cs.drjava.model.compiler.JavacCompiler.runCommand(JavacCompiler.java:272)
```

Stack Trace



```
java.lang.StringIndexOutOfBoundsException: String index out of range: 100
	at java.lang.String.charAt(String.java:658)
	at ErrorTest.main(ErrorTest.java:8)
	at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
	at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
	at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
	at java.lang.reflect.Method.invoke(Method.java:498)
	at edu.rice.cs.drjava.model.compiler.JavacCompiler.runCommand(JavacCompiler.java:272)
```

- Luckily, Java will give you a **stack trace** where the error occurred
- Every stack trace is different, but in this one:
- The top line is the precise error: `StringIndexOutOfBoundsException`
- The next line is the spot in Java code where the error occurred
- The next line is where in your code the error happened - at `ErrorTest.main(ErrorTest.java:8)`