CSC 212 PROGRAMMING WITH DATA STRUCTURES

SPRING 2016

PROF. SARA SHEEHAN

SMITH COLLEGE

CLASS 11: MARCH 1 OUTLINE

- Recap MSA + pair programming
- Debrief Homework 4
- Handling exceptions (Lab 5)
- Casting, Static, Iterators revisited
- Built-in data structures
- Begin 4rd data structure: Queues
- Reminder: .equals() for strings

MSA

- Thank you for all the feedback!
- Keep doing: board photos, mix of things during lecture

MSA

- Thank you for all the feedback!
- Keep doing: board photos, mix of things during lecture

- Did I miss a prereq?
- More emphasis on syntax and foundational material

MSA

- Thank you for all the feedback!
- Keep doing: board photos, mix of things during lecture

- Did I miss a prereq?
- More emphasis on syntax and foundational material

- Solutions

MSA

- Thank you for all the feedback!
- Keep doing: board photos, mix of things during lecture

- Did I miss a prereq?
- More emphasis on syntax and foundational material

- Solutions

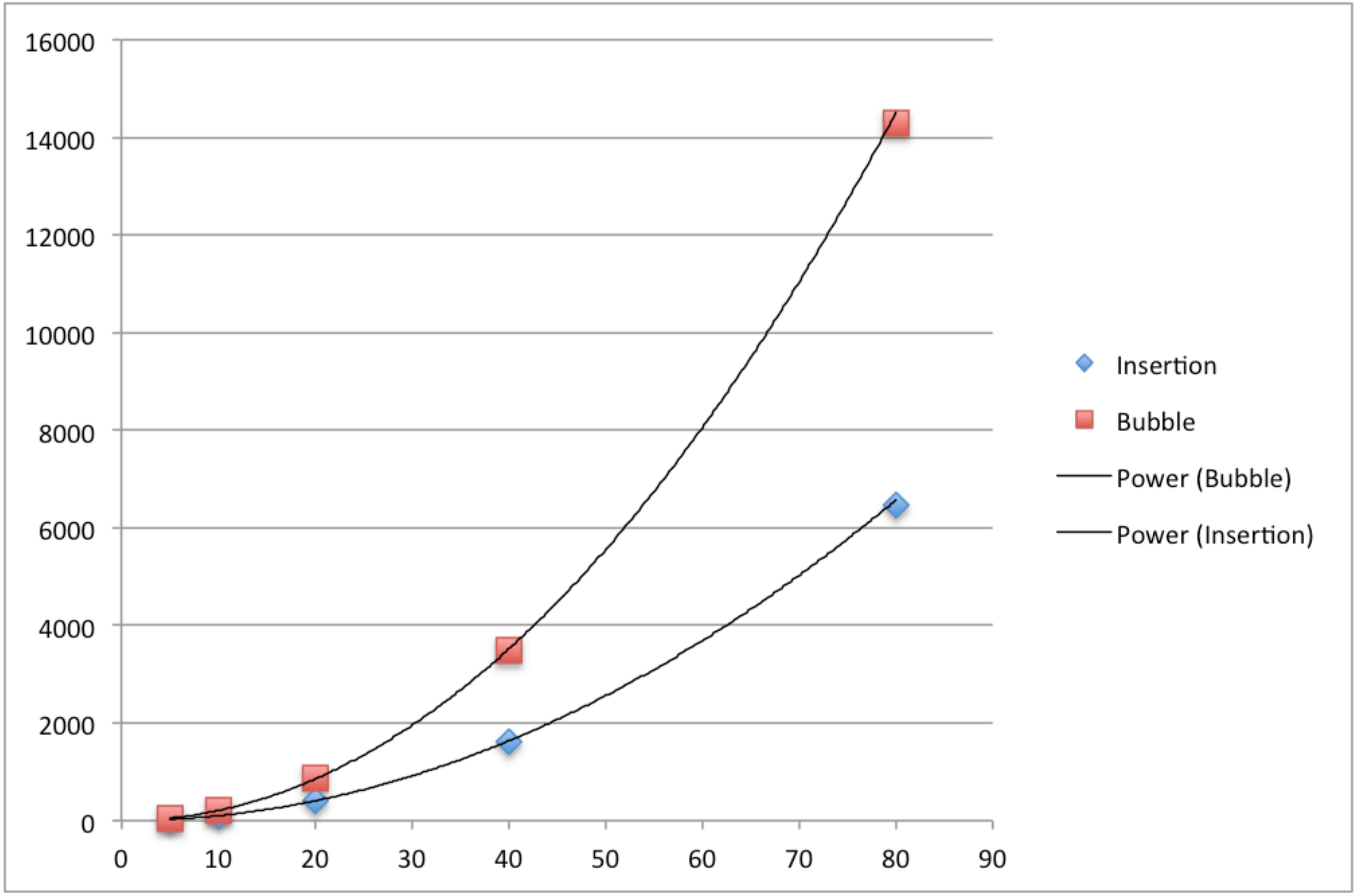
- Translating theory into code

WHAT YOU SAID ABOUT IMPROVING LEARNING

- Start homework earlier (a bit each day)
- Read the textbooks
- Read Javadocs
- Ask more questions in class
- Go to office hours and TA hours
- Revisit labs, homeworks, and lectures
- Stack overflow
- Collaborate with classmates
- Piazza

I would add: use pencil and paper more

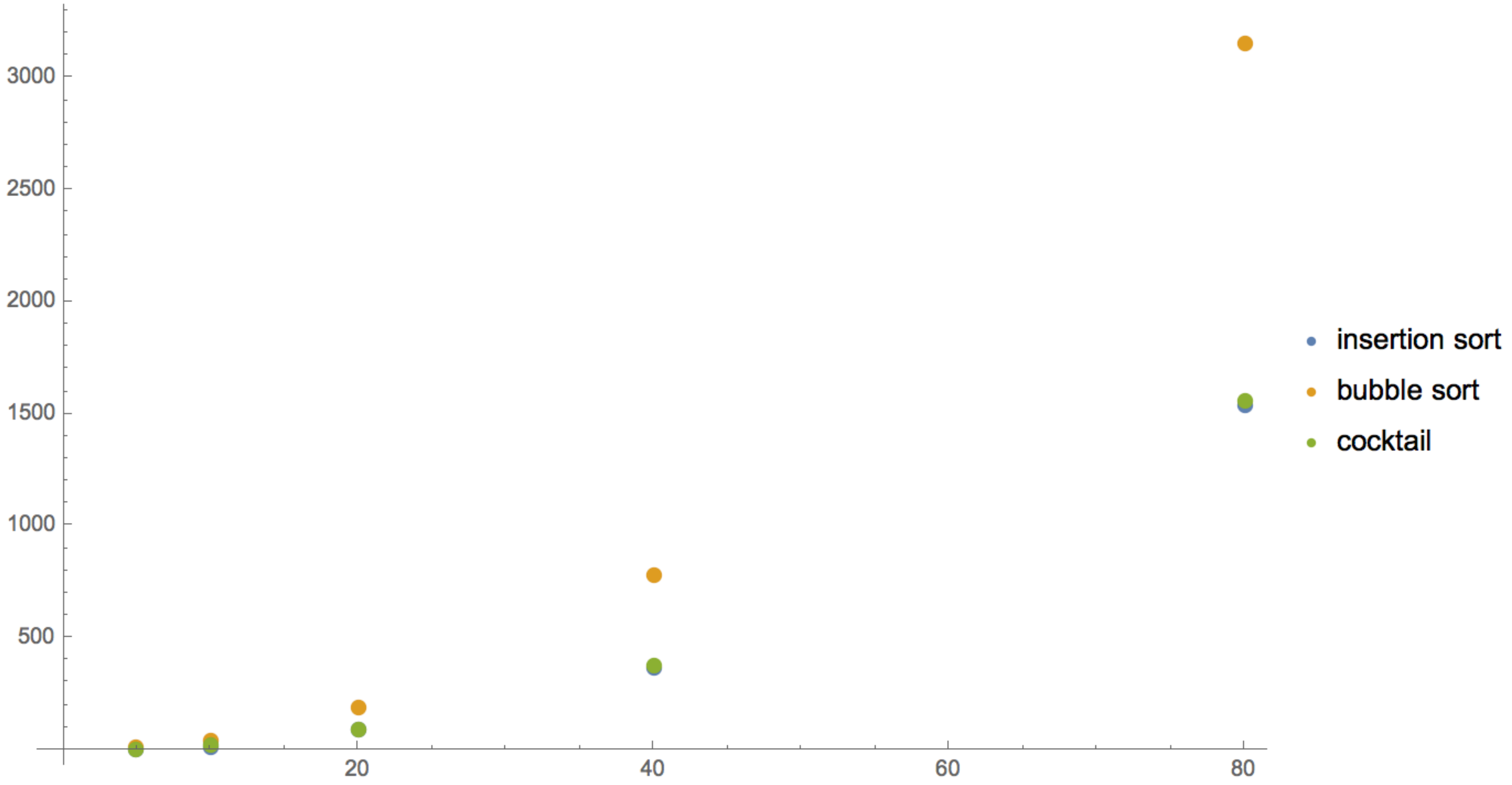
HOMEWORK 4



Credit: Margaret and Serena

HOMEWORK 4

```
insertion = {{5, 3}, {10, 12}, {20, 95}, {40, 370}, {80, 1546}};
bubble = {{5, 10}, {10, 45}, {20, 190}, {40, 780}, {80, 3160}};
cocktail = {{5, 4}, {10, 20}, {20, 90}, {40, 380}, {80, 1560}};
ListPlot[{insertion, bubble, cocktail},
 PlotLegends → {"insertion sort", "bubble sort", "cocktail"}]
```

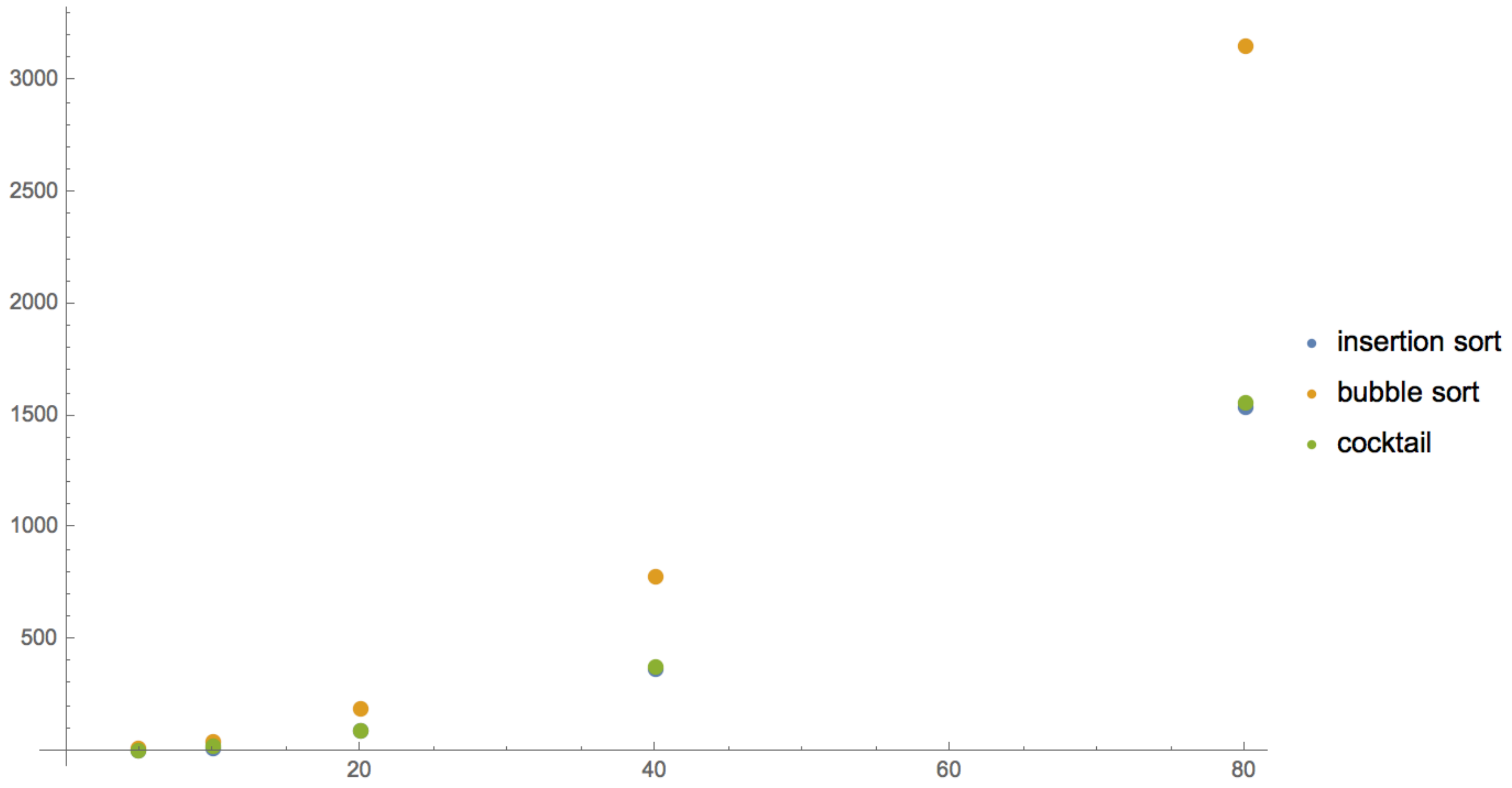


Credit: Claudia

HOMEWORK 4

= 80*79/2

```
insertion = {{5, 3}, {10, 12}, {20, 95}, {40, 370}, {80, 1546}};
bubble = {{5, 10}, {10, 45}, {20, 190}, {40, 780}, {80, 3160}};
cocktail = {{5, 4}, {10, 20}, {20, 90}, {40, 380}, {80, 1560}};
ListPlot[{insertion, bubble, cocktail},
 PlotLegends → {"insertion sort", "bubble sort", "cocktail"}]
```



Credit: Claudia

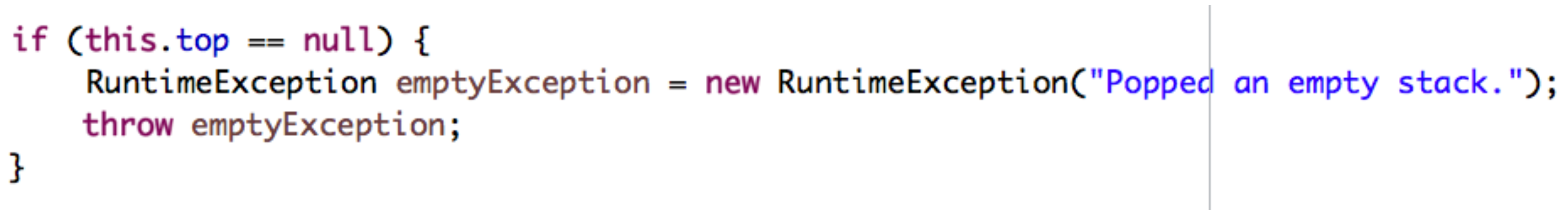
EXCEPTIONS (LAB 5)

Inside Stack

```
if (this.top == null) {
    RuntimeException emptyException = new RuntimeException("Popped an empty stack.");
    throw emptyException;
}
```

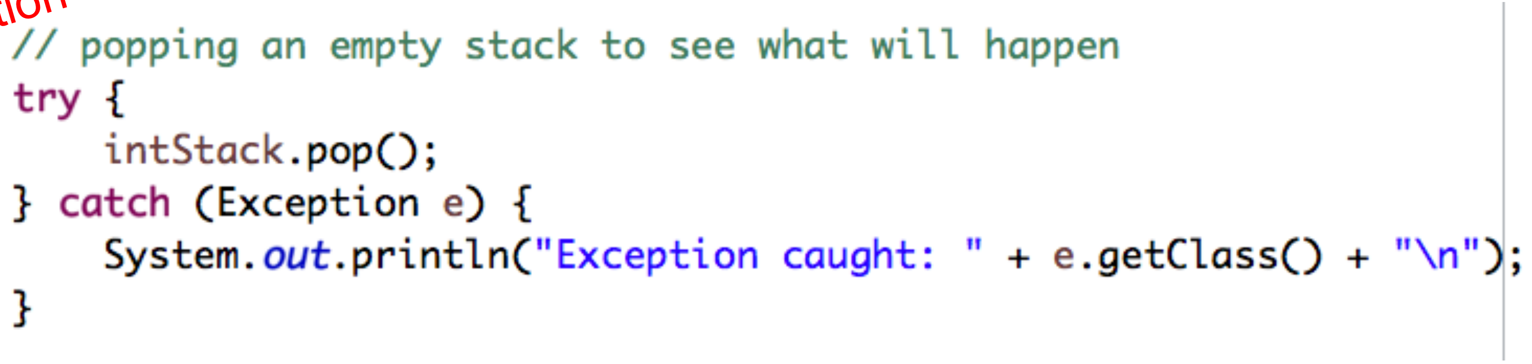
EXCEPTIONS (LAB 5)

Inside Stack



```java
if (this.top == null) {
    RuntimeException emptyException = new RuntimeException("Popped an empty stack.");
    throw emptyException;
}
```

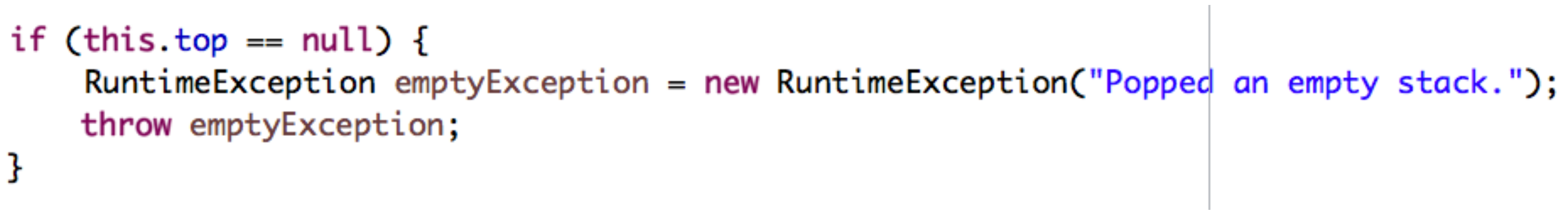
Option 1



```java
// popping an empty stack to see what will happen
try {
    intStack.pop();
} catch (Exception e) {
    System.out.println("Exception caught: " + e.getClass() + "\n");
}
```

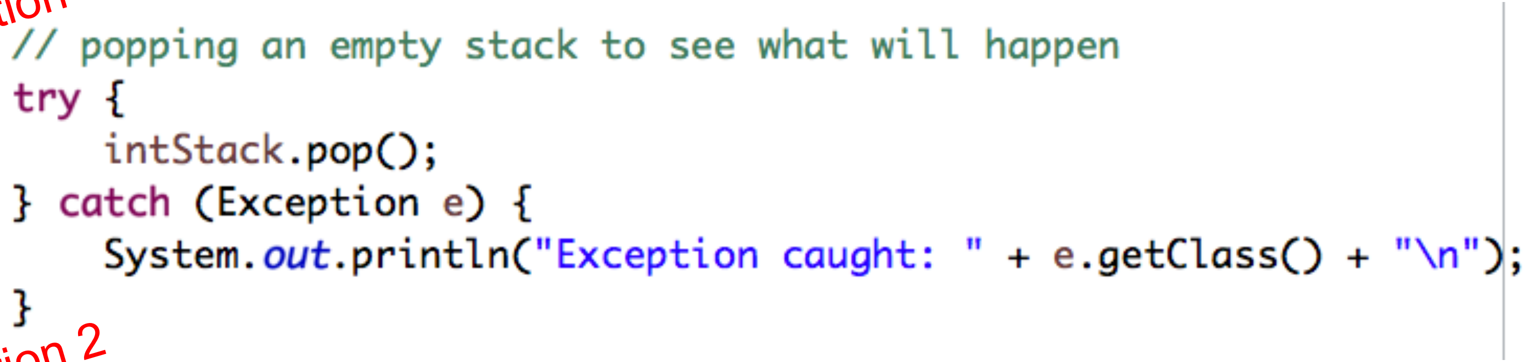
EXCEPTIONS (LAB 5)

Inside Stack



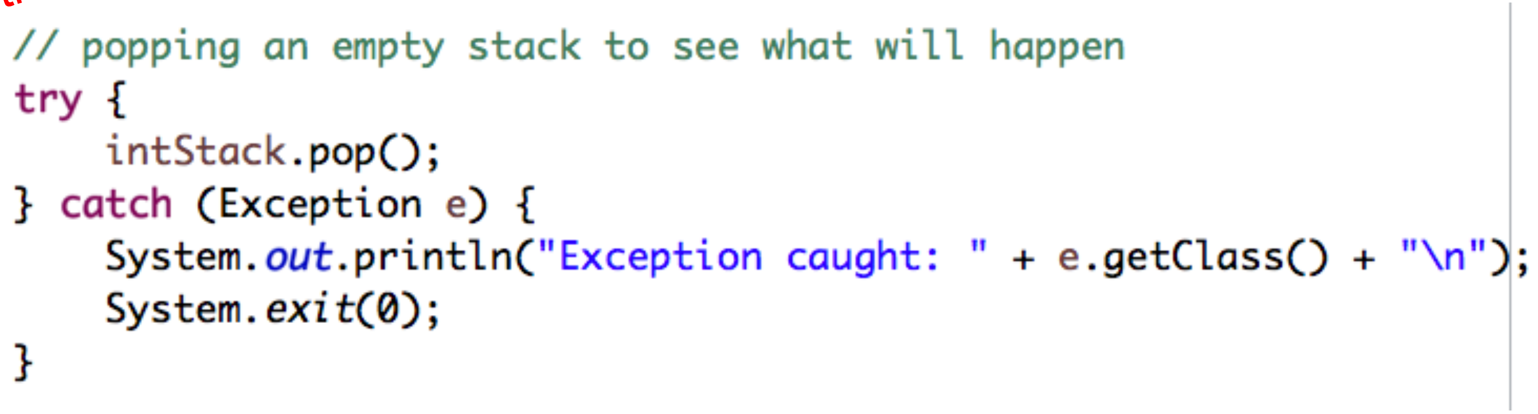
```
if (this.top == null) {
    RuntimeException emptyException = new RuntimeException("Popped an empty stack.");
    throw emptyException;
}
```

Option 1



```
// popping an empty stack to see what will happen
try {
    intStack.pop();
} catch (Exception e) {
    System.out.println("Exception caught: " + e.getClass() + "\n");
}
```

Option 2



```
// popping an empty stack to see what will happen
try {
    intStack.pop();
} catch (Exception e) {
    System.out.println("Exception caught: " + e.getClass() + "\n");
    System.exit(0);
}
```

EXCEPTIONS (LAB 5)

Inside Stack

```
if (this.top == null) {
    RuntimeException emptyException = new RuntimeException("Popped an empty stack.");
    throw emptyException;
}
```

Option 1

```
// popping an empty stack to see what will happen
try {
    intStack.pop();
} catch (Exception e) {
    System.out.println("Exception caught: " + e.getClass() + "\n");
}
```

Option 2

```
// popping an empty stack to see what will happen
try {
    intStack.pop();
} catch (Exception e) {
    System.out.println("Exception caught: " + e.getClass() + "\n");
    System.exit(0);
}
```

Option 3: do nothing

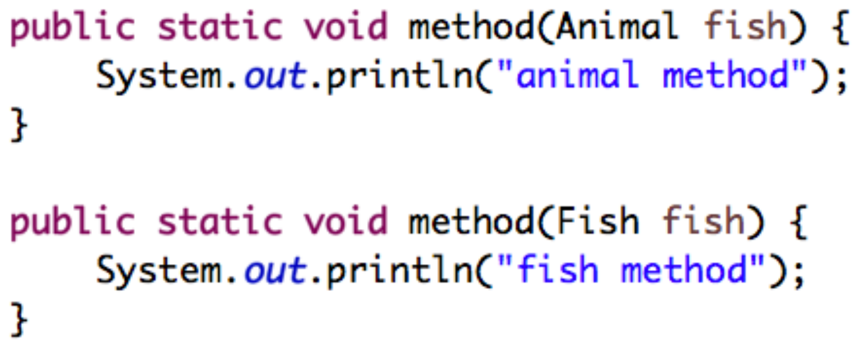
CASTING (DOWN)

```
List<String> myList;
if (isArrayList) {
    myList = new ArrayList<String>();
} else {
    myList = new LinkedList<String>();
}
```

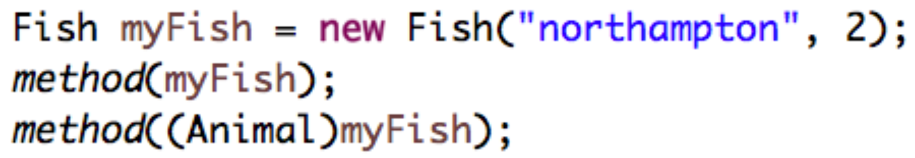
Downcasting

```
ArrayList<String> anotherList = (ArrayList<String>) myList;
```

CASTING (UP)



```java
public static void method(Animal fish) {
    System.out.println("animal method");
}

public static void method(Fish fish) {
    System.out.println("fish method");
}
```



```java
Fish myFish = new Fish("northampton", 2);
method(myFish);
method((Animal)myFish);
```

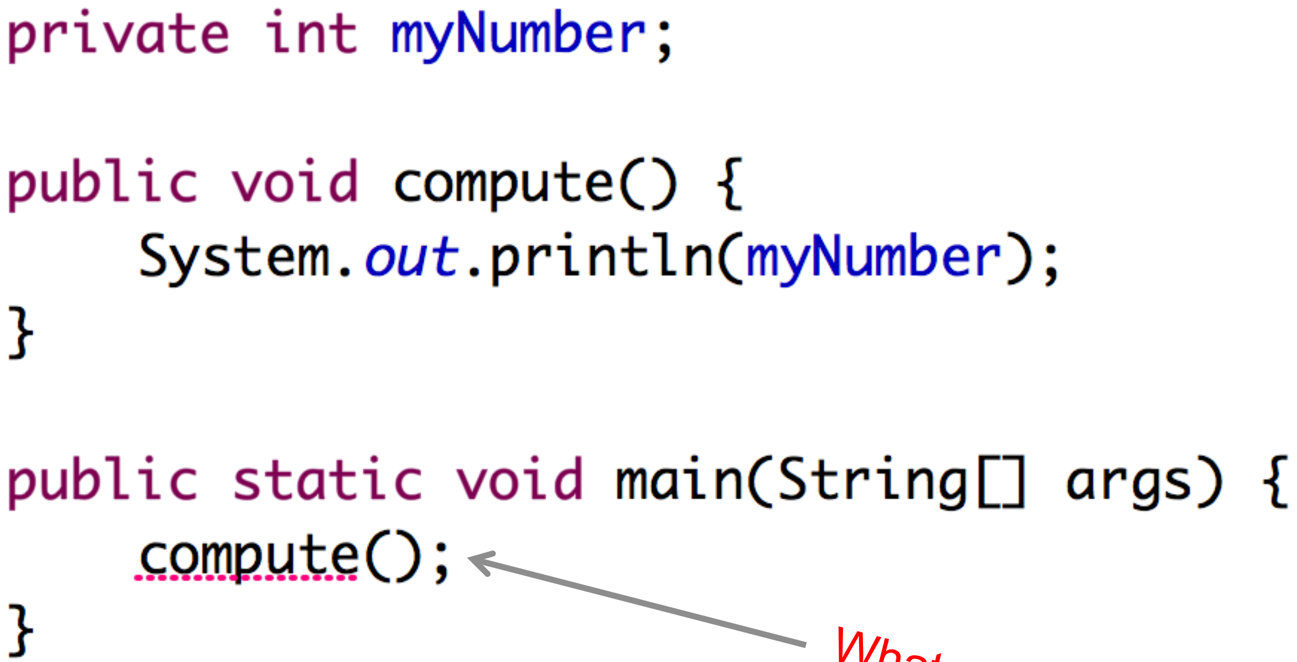
CASTING (UP)

```java
public static void method(Animal fish) {
    System.out.println("animal method");
}

public static void method(Fish fish) {
    System.out.println("fish method");
}
```

```java
Fish myFish = new Fish("northampton", 2);
method(myFish);
method((Animal)myFish);
```

Prints:
fish method
animal method

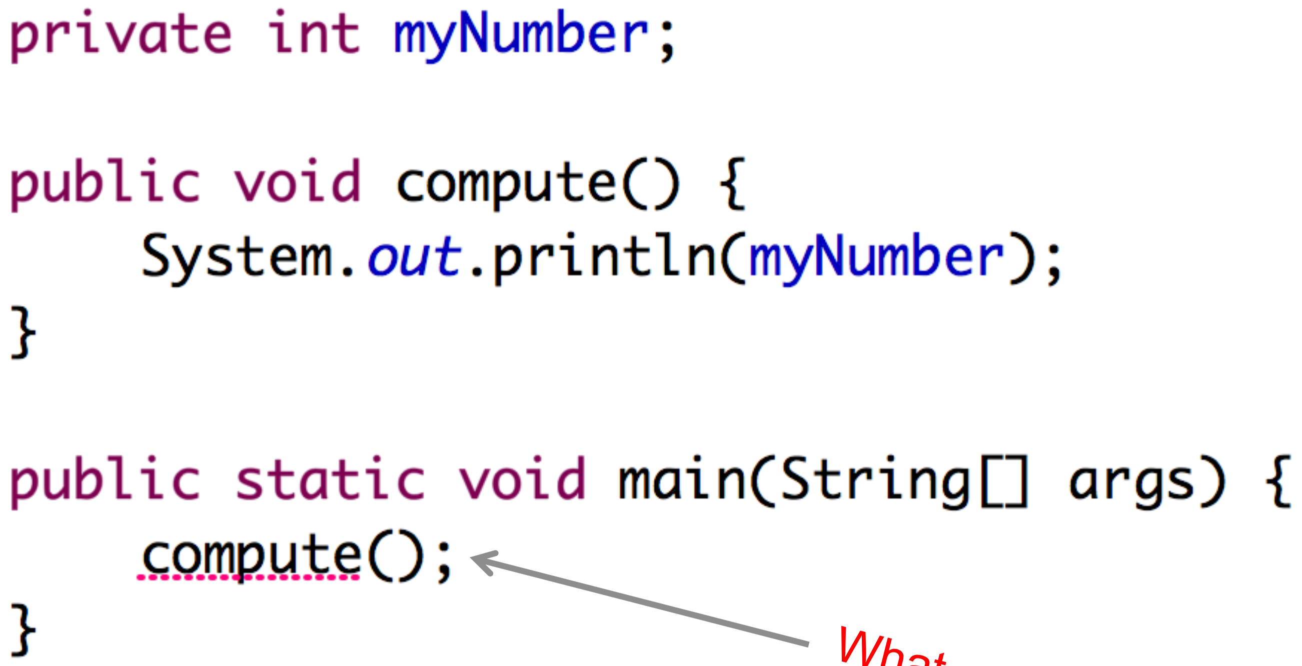
STATIC METHODS



```
private int myNumber;

public void compute() {
    System.out.println(myNumber);
}

public static void main(String[] args) {
    compute();
}
```

What error does this give?

STATIC METHODS

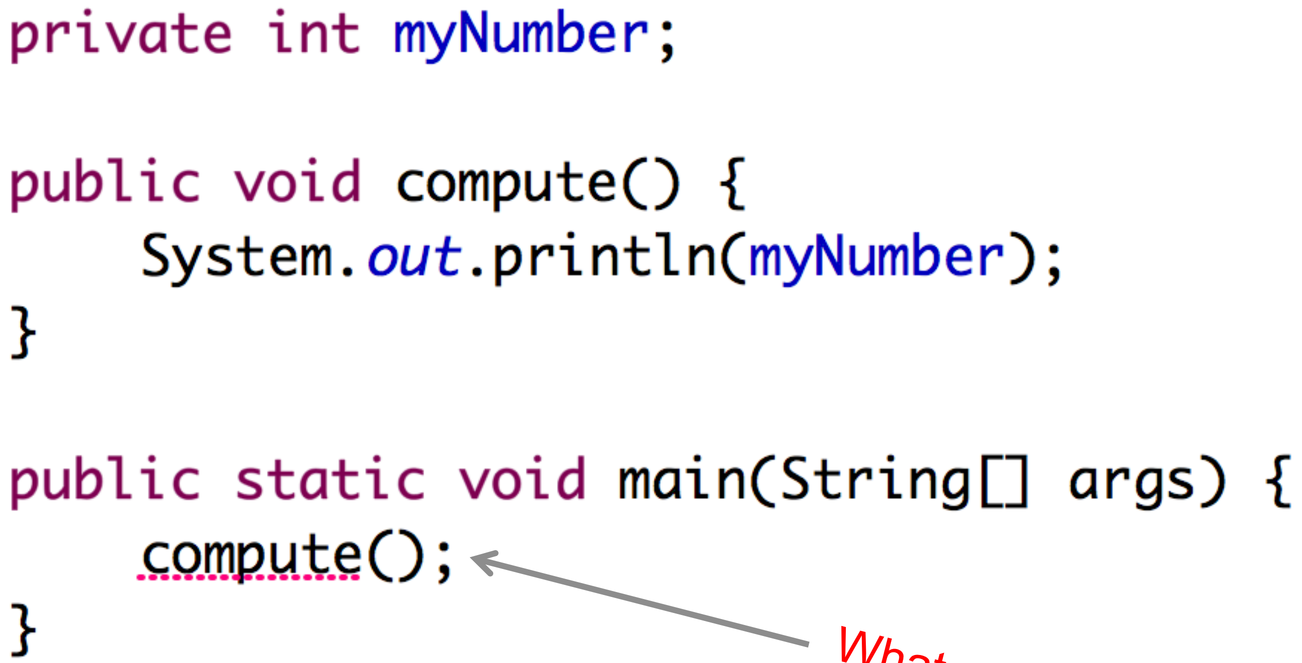


```
private int myNumber;

public void compute() {
    System.out.println(myNumber);
}

public static void main(String[] args) {
    compute();
}
```

What error does this give?

"Cannot make a static reference to a non-static method."
Why??

STATIC METHODS



```
private int myNumber;

public void compute() {
    System.out.println(myNumber);
}

public static void main(String[] args) {
    compute();
}
```

What error does this give?

"Cannot make a static reference to a non-static method."
Why??

Solution: both compute() and myNumber need to be static

ITERATORS

```
LinkedList<String> courses = new LinkedList<String>();
courses.add("CSC 212");
courses.add("CSC 102");
courses.add("CSC 111");

Iterator<String> courseIterator = courses.iterator();
while (courseIterator.hasNext()) {
    System.out.println(courseIterator.next());
}
System.out.println();

for (String s : courses) {
    System.out.println(s);
}
```

ITERATORS

```
LinkedList<String> courses = new LinkedList<String>();
courses.add("CSC 212");
courses.add("CSC 102");
courses.add("CSC 111");

Iterator<String> courseIterator = courses.iterator();
while (courseIterator.hasNext()) {
    System.out.println(courseIterator.next());
}
System.out.println();
```

```
for (String s : courses) {
    System.out.println(s);
}
```

foreach loop