MODULE 1: INTRODUCTION TO PROGRAMMING

Introduction to Collections (Part 2)

TECH ELEVATOR
A Stride Company

Yesterday

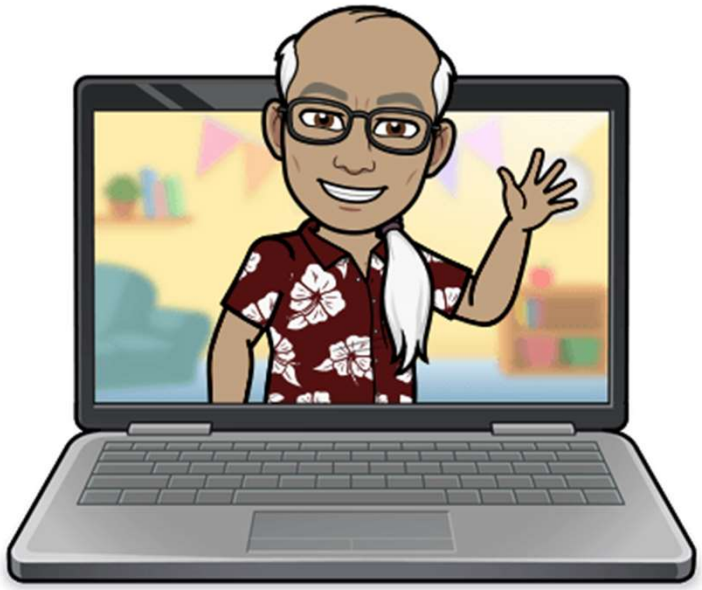
- What is a **package**?
- What are some properties of a **list**?
- What are some properties of a **stack**?
- What are some properties of a **queue**?

Collections: Map<T,T>

- A **map** is an indexed collection that allows values to be located using user-defined keys.
- **You** define the data types of the key and values
- Keys **must** be unique
- Order is not guaranteed

87 => Sydney Crosby
59 => Jake Guentzel
71 => Evgeni Malkin
17 => Bryan Rust

LET’S CODE!



Collections: LinkedHashMap<T,T>

- A **map** is an indexed collection that allows values to be located using user-defined keys.
- **You** define the data types of the key and values
- Keys **must** be unique
- **Order** is maintained

71 => Evgeni Malkin
17 => Bryan Rust
59 => Jake Guentzel
87 => Sydney Crosby

Collections: TreeMap<T,T>

- A map is an indexed collection that allows values to be located using user-defined keys.
- You define the data types of the key and values
- Keys must be unique
- Keys are **sorted**

17 => Bryan Rust

59 => Jake Guentzel

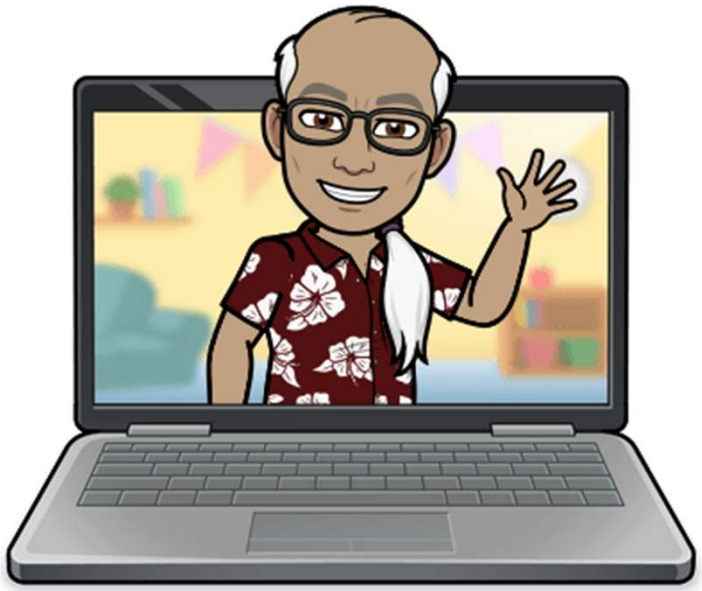
71 => Evgeni Malkin

87 => Sydney Crosby

Collections: HashSet<T>

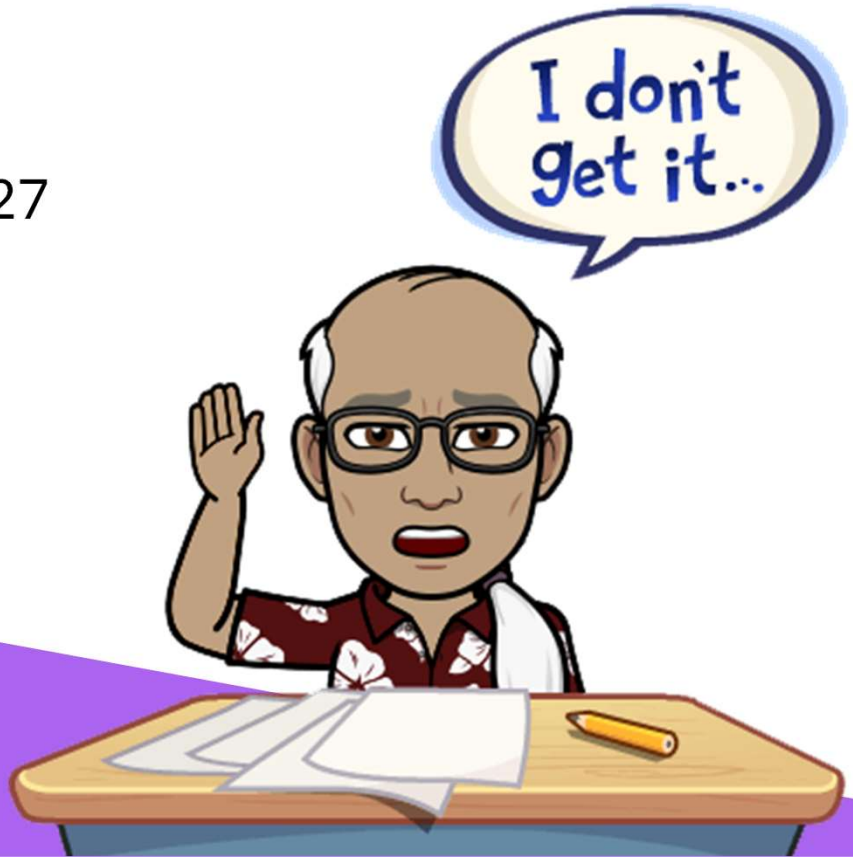
- A **HashSet** is like a list except it does not allow duplicates.
- Elements are not kept in order

LET’S CODE!



The Problem

```
double d1 = 3.47;
double d2 = 3.17;
// prints 0.30000000000000027
System.out.println(d1 - d2);
```



The Solution

- BigDecimal
  - A very precise object to handle floating point math.
  - As an object, needs instantiated and manipulated with methods
  - Immutable

```
BigDecimal thingOne = new BigDecimal(“3.47”);
BigDecimal thingTwo = new BigDecimal(“3.17”);
```

Math with BigDecimal

- BigDecimal thingOne = new BigDecimal(“3.47”);
- BigDecimal thingTwo = new BigDecimal(“3.17”);
- BigDecimal sum = thingOne.add(thingTwo);
- BigDecimal difference = thingOne.subtract(thingTwo);
- BigDecimal product = thingOne.multiply(thingTwo);
- BidDecimal ratio = thingOne.divide(thingTwo);

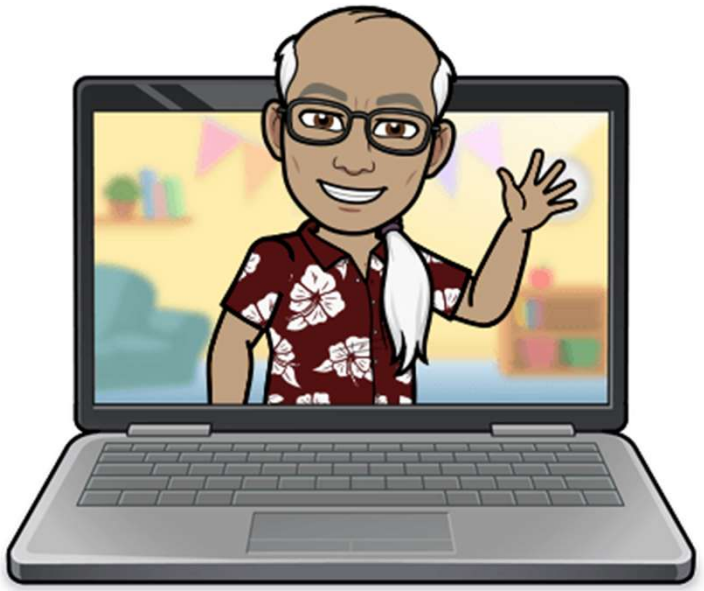
Comparison

- Can’t use ==,>,>=,<,<=
- BigDecimal is an object, so use method
  - compareTo()

```
BigDecimal thingOne = new BigDecimal(“3.47”);
BigDecimal thingTwo = new BigDecimal(“3.17”);
BigDecimal thingThre = new BigDecimal(“3.17”);

int oneVersusTwo = thingOne.compareTo(thingTwo);          thingOne > thingTwo => 1
int twoVersusOne = thingTwo.compareTo(thingOne);          thingTwo < thingOne => -1
int twoVersusThree = thingTwo.compareTo(thingThree);      thingTwo == thingThree => 0
```

LET’S CODE!



WHAT QUESTIONS DO YOU HAVE?

Code Reviews

- 5 minutes
- Sign up for slots
- Make sure no Pathway conflicts

Reading for tonight:

Classes and Encapsulation