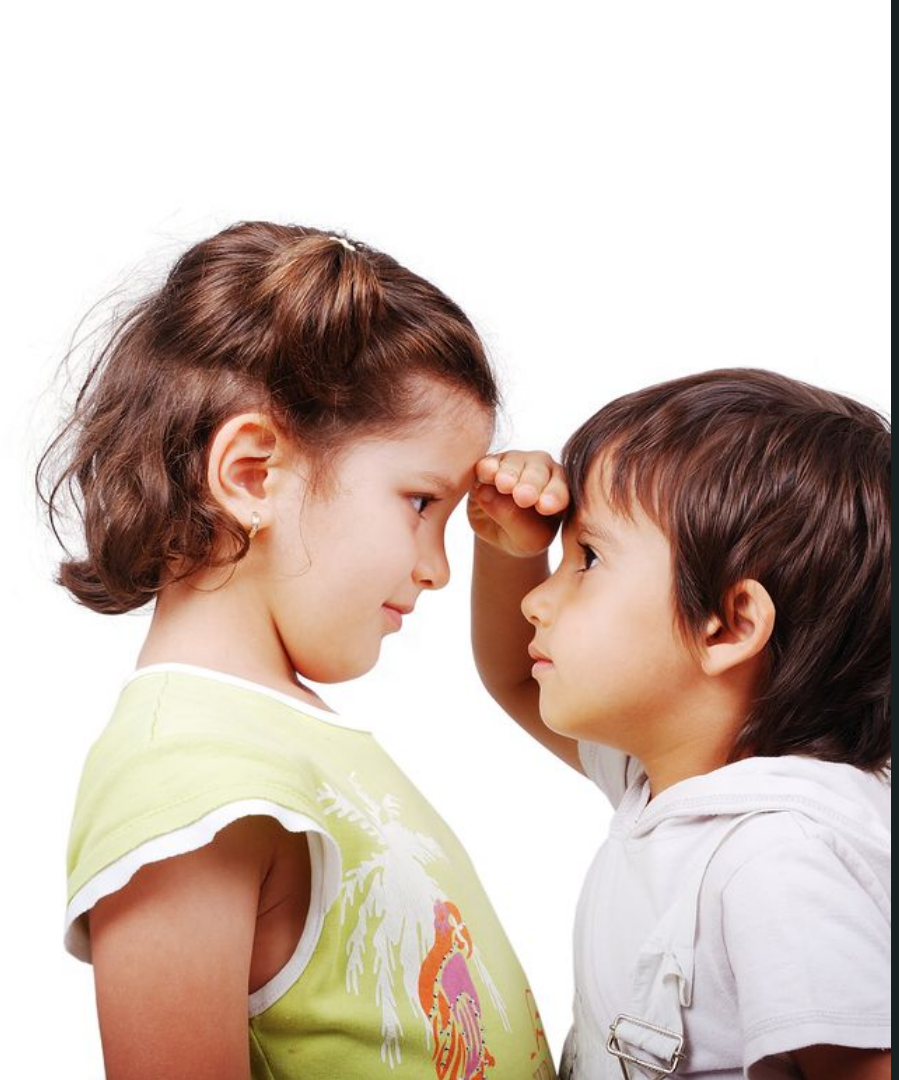

Level up
your basic Java
understanding

Lilian Cavalet
Master’s degree in Applied Mathematics
Software developer
Foodie
Cat lover

Alan Bastos
Master's degree in Psychology
Java Developer
Gamer
Fantasy reader


CHALLENGING WAY
RIGHT WAY
HARD WAY
FAST WAY
DIFFICULT WAY
EASY WAY

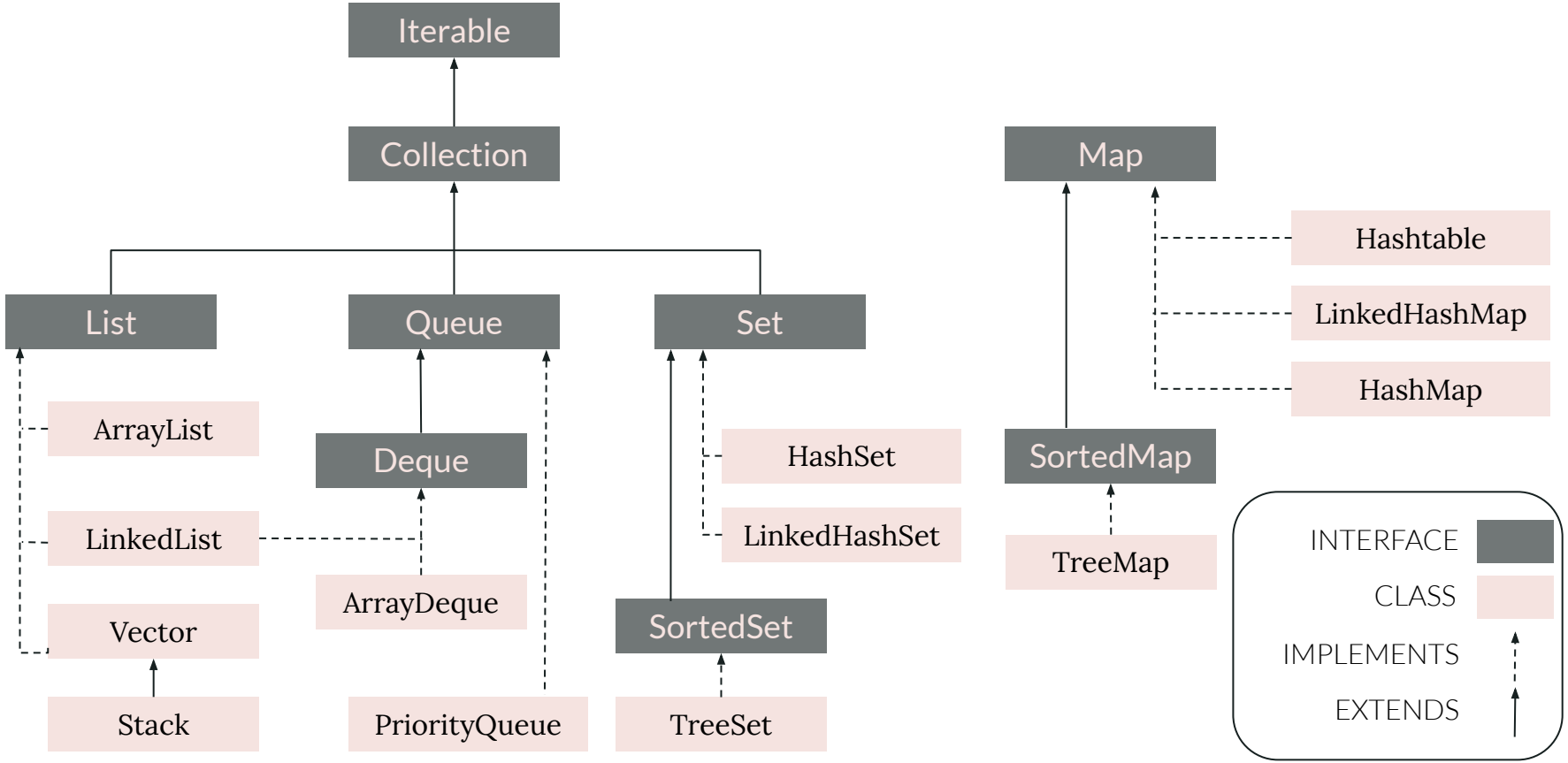

collections
framework

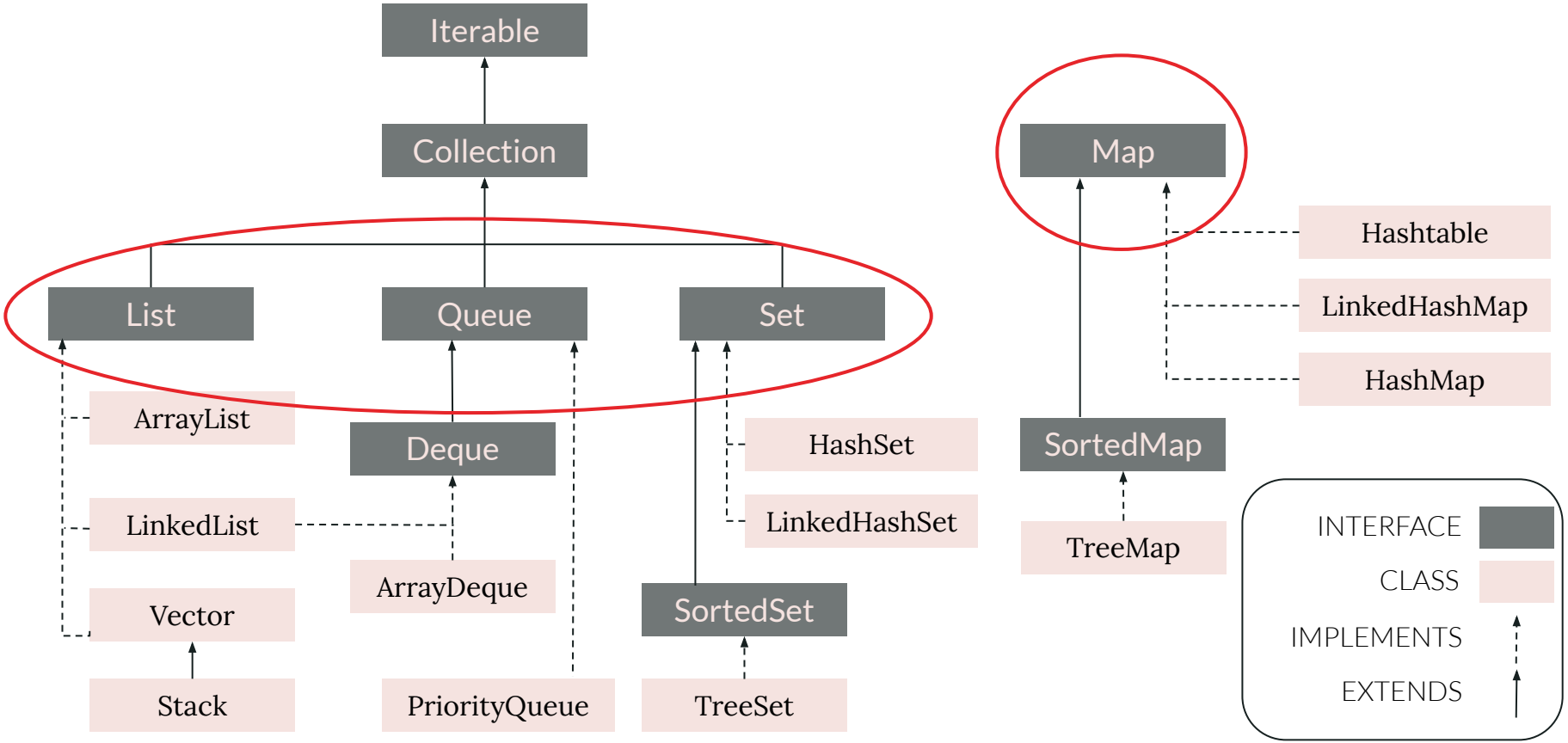


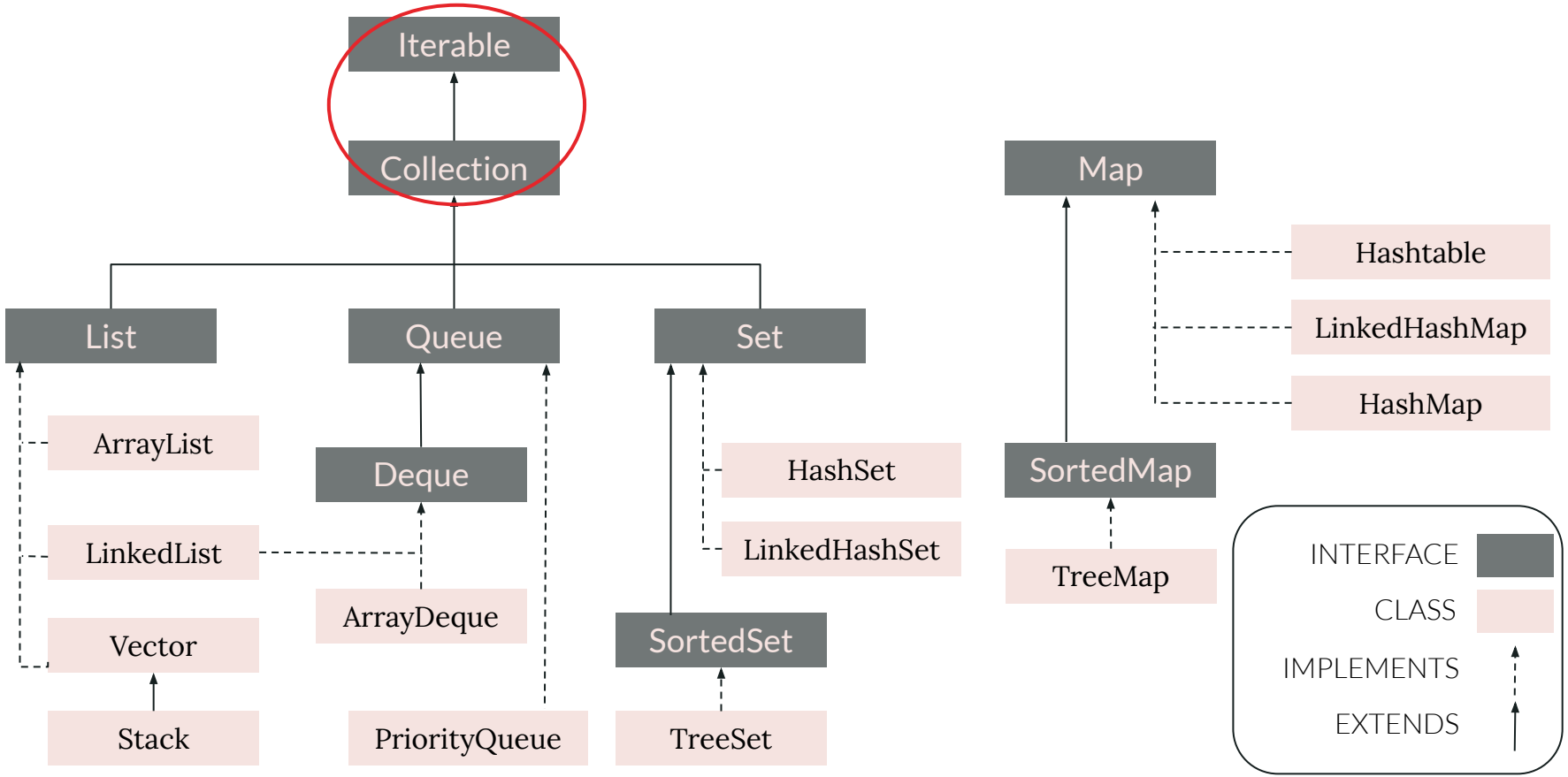
```
int childrensHeights[] = { 95, 103, 147, 110 };
int arrayLength = 4;
int tallest = childrensHeights[0];

for (int i = 0; i < arrayLength; i++) {
        if (tallest < childrensHeights[i]) {
                tallest = childrensHeights[i];
        }
}
System.out.print(tallest); //147
```


how does
the collections
framework
works?


Iterable
Collection
List
Queue
Set
ArrayList
LinkedList
Vector
Stack
Deque
ArrayDeque
PriorityQueue
HashSet
LinkedHashSet
SortedSet
TreeSet
Map
Hashtable
LinkedHashMap
HashMap
SortedMap
TreeMap
INTERFACE
CLASS
IMPLEMENTS
EXTENDS


Iterable
Collection
List
Queue
Set
ArrayList
Deque
HashSet
LinkedList
LinkedHashSet
ArrayDeque
Vector
SortedSet
Stack
PriorityQueue
TreeSet
Map
Hashtable
LinkedHashMap
HashMap
SortedMap
TreeMap
INTERFACE
CLASS
IMPLEMENTS
EXTENDS


Iterable
Collection
List
Queue
Set
ArrayList
LinkedList
Vector
Stack
Deque
ArrayDeque
PriorityQueue
HashSet
LinkedHashSet
SortedSet
TreeSet
Map
Hashtable
LinkedHashMap
HashMap
SortedMap
TreeMap
INTERFACE
CLASS
IMPLEMENTS
EXTENDS

```java
List<Integer> childrensHeights = new ArrayList<>();
childrensHeights.add(95);
childrensHeights.add(103);
Collections.addAll(childrensHeights, 147, 110);
int tallest = childrensHeights.get(0);
for (Integer height : childrensHeights) {
        if (tallest < height) {
                tallest = height;
        }
}
System.out.print(tallest); //147
```

```
List<Integer> childrensHeights = new ArrayList<>();
childrensHeights.add(95);
childrensHeights.add(103);
Collections.addAll(childrensHeights, 147, 110);
int tallest = childrensHeights.get(0);
for (Integer height : childrensHeights) {
        if (tallest < height) {
                tallest = height;
        }
}
System.out.print(tallest); //147
```

```java
List<Integer> childrensHeights = new ArrayList<>();
childrensHeights.add(95);
childrensHeights.add(103);
Collections.addAll(childrensHeights, 147, 110);
int tallest = childrensHeights.get(0);
for (Integer height : childrensHeights) {
        if (tallest < height) {
                tallest = height;
        }
}
System.out.print(tallest); //147
```

```java
List<Integer> childrensHeights = new ArrayList<>();
childrensHeights.add(95);
childrensHeights.add(103);
Collections.addAll(childrensHeights, 147, 110);
int tallest = childrensHeights.get(0);
for (Integer height : childrensHeights) {
        if (tallest < height) {
                tallest = height;
        }
}
System.out.print(tallest); //147
```

```java
List<Integer> childrensHeights = new ArrayList<>();
childrensHeights.add(95);
childrensHeights.add(103);
Collections.addAll(childrensHeights, 147, 110);
int tallest = childrensHeights.stream()
                .mapToInt(v -> v).max()
                .orElseThrow(NoSuchElementException::new);
System.out.print(tallest); // 147
```

```
List<Integer> childrensHeights = new ArrayList<>();
childrensHeights.add(95);
childrensHeights.add(103);
Collections.addAll(childrensHeights, 147, 110);
List<Integer> childrensHeightsUnmodifiableList =
Collections.unmodifiableList(childrensHeights);

childrensHeightsUnmodifiableList.add(111) ;  //Exception
childrensHeights.add(111);
System.out.print(childrensHeights); //[95, 103, 147, 110, 111]
```

```
List<Integer> childrensHeightsViaArrayAsList = Arrays.asList(95, 103,147, 110);
childrensHeightsViaArrayAsList.add(111); //Exception

List<Integer> childrensHeightsViaArrayList = new ArrayList<>(Arrays.asList(95,
                                                                103, 147, 110));
childrensHeightsViaArrayList.add(111);
System.out.print(childrensHeights); //[95, 103, 147, 110, 111]
```

```
Person randomPerson = new Person("John Doe", 42);
List<Object> listWithMiscObjects = Arrays.asList("TDC", 2021, randomPerson);
System.out.println(listWithMiscObjects);
 //[TDC, 2021, Person [age=42, name=John Doe]]
```

GENERICS

```java
List<Integer> childrensHeights = new ArrayList<>();
childrensHeights.add(95);
childrensHeights.add(103);
Collections.addAll(childrensHeights, 147, 110);
int tallest = childrensHeights.get(0);
for (Integer height : childrensHeights) {
        if (tallest < height) {
                tallest = height;
        }
}
System.out.print(tallest); //147
```

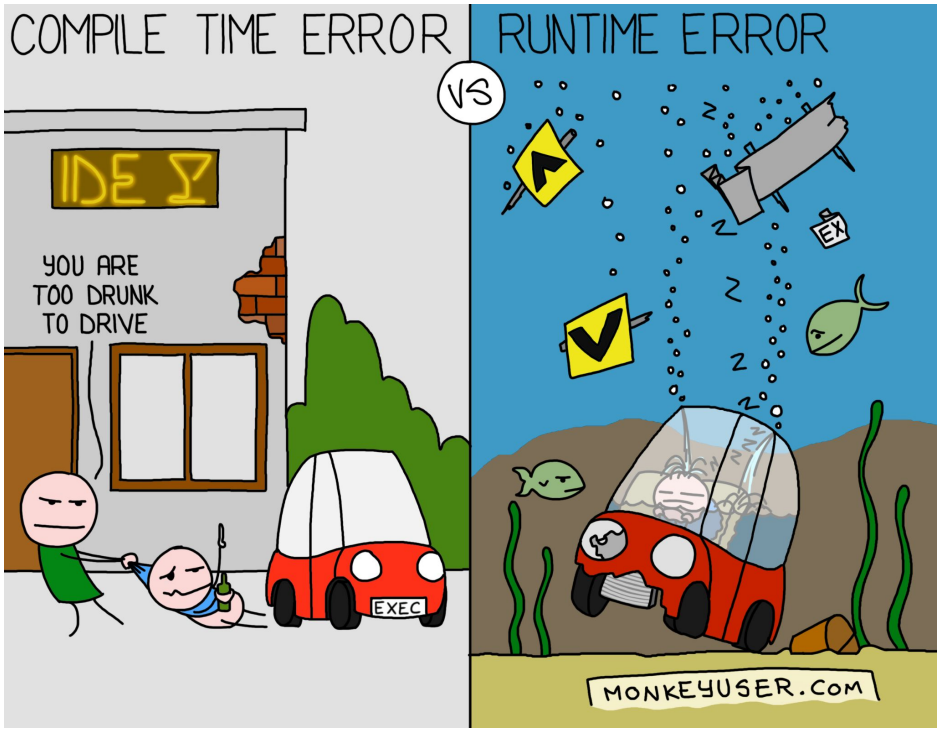
```java
List<Integer> childrensHeights = new ArrayList<>();
childrensHeights.add(95);
childrensHeights.add(103);
Collections.addAll(childrensHeights, 147, 110);
int tallest = childrensHeights.get(0);
for (Integer height : childrensHeights) {
        if (tallest < height) {
                tallest = height;
        }
}
System.out.print(tallest); //147
```

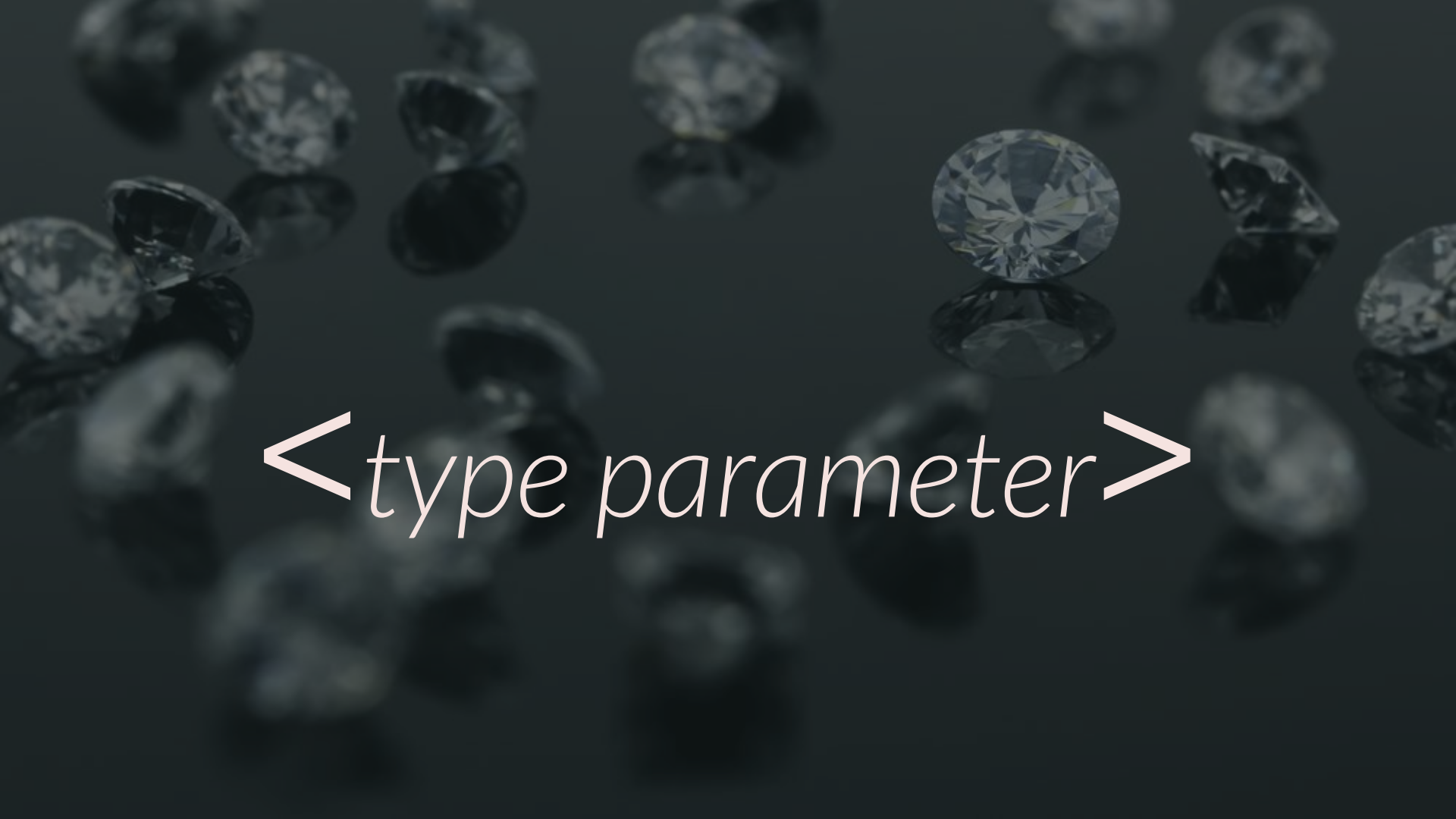
```java
List<Integer> childrensHeights = new ArrayList<>();
childrensHeights.add(95);
childrensHeights.add(103);
Collections.addAll(childrensHeights, 147, 110);
int tallest = childrensHeights.get(0);
for (Integer height : childrensHeights) {
        if (tallest < height) {
                tallest = height;
        }
}
System.out.print(tallest); //147
```

```
List childrensHeights = new ArrayList();
childrensHeights.add(95);
childrensHeights.add(103);
Collections.addAll(childrensHeights, 147, 110);
int tallest = (Integer) childrensHeights.get(0);
for(Object height : childrensHeights) {
   if (tallest < (Integer) height) {
      tallest = (Integer) height;
   }
}
System.out.print(tallest); // 147
```

```
List childrensHeights = new ArrayList();
childrensHeights.add(95);
childrensHeights.add(103);
Collections.addAll(childrensHeights, 147, 110);
int tallest = (Integer) childrensHeights.get(0);
for(Object height : childrensHeights) {
   if (tallest < (Integer) height) {
      tallest = (Integer) height;
   }
}
System.out.print(tallest); // 147
```

How **Generics** works?




<type parameter>

type erasure

```
Integer[] childrensHeights = { 95, 103, 147, 110 };

public static <T> List<T> fromArrayToList(T[] a) {
        return Arrays.stream(a).collect(Collectors.toList()); //[95, 103, 147, 110]
}
```

```
Child child1 = new Child("John Doe", 95);
Child child2 = new Child("Jane Doe", 103);
Child child3 = new Child("Richard Roe", 147);
Child children[] = { child1, child2, child3 };

public static <T extends Person> List<T> fromArrayToList(T[] a) {
        return Arrays.stream(a).collect(Collectors.toList());
        //[Child [height=95, name=John Doe],
        //Child [height=103, name=Jane Doe],
        //Child [height=147, name=Richard Roe]]
}
```

```
List<?> wildcards = new ArrayList<>();
wildcards.add(null);
System.out.println(wildcards); //[null]
```

```
Child child1 = new Child("Jhon Doe", 95);
Child child2 = new Child("Jane Doe", 103);
Child child3 = new Child("Richard Roe", 147);
List<Child> children = new ArrayList<>();
Collections.addAll(children, child1, child2, child3);

public static Person findTallest(final List<? extends Person> people){
        return people.stream().max(Comparator.comparingInt(Person::getHeight))
                              .orElseThrow(NoSuchElementException::new);
                             //Child [height=147, name=Richard Roe]
}
```


REWIND


Savings


ALWAYS REMEMBER...
SAFETY FIRST!

Q & A

The code used in this presentation is available here

https://git.io/JZtXa