Understanding Generics

Dan Bunker
Consultant | Saas Builder | Mentor

linkedin.com/in/bunkerdan

```
List numbers = new ArrayList();
numbers.add(1);   //Integer
numbers.add(2L);  //Long
numbers.add("3"); //String
numbers.add(4.0); //Double

for(int i = 0; i < numbers.size(); i++) {
    int myNumber = (int)numbers.get(i);
}

List<Integer> numbers = new ArrayList<>();
numbers.add(1);   //Integer
numbers.add(2L);  //Long - Compile Error
numbers.add("3"); //String - Compile Error
numbers.add(4.0); //Double - Compile Error
```

Generic Naming Conventions

E–Element (used extensively by the Java Collections Framework)

K–Key

N–Number

T–Type

V–Value

S, U, V, and so on–Second, third, and fourth types

Generic Wildcards and Bounds

```
public void printList(List<?> list){
    list.forEach(System.out::println);
}
```

```
public void printList(List<? extends String> list){
    list.forEach(System.out::println);
}
```

```
public void printList(List<? super String> list){
    list.forEach(System.out::println);
}
```

Summary

- Java generics naming conventions
- Class generic types <Type>
- Method generics
- Generics support wildcards <?>, extends, and super bounding
- Autoboxing - primitive <-> Wrapper Class

Final Exam Review Concepts

Analyze and understand code involving Array and ArrayList

Understand the Collection API hierarchy and what each collection type does

Collection API common methods

Comparable/Comparator

Autoboxing and unboxing

Generics