

## Generic Types

- Example:
  - ArrayList< BankAccount >
  - Set< String >
- Benefit
  - no casting needed
  - no risk of runtime exceptions
- Defining a generic type
  - use type parameter like any other type name
  - cannot be used in: extends, new, static fields and methods
- Constraining type variables
  - class Calculator <E extends Number>

## Generic Types and Methods

- Virtual machine does not support generic types
  - all handled by compiler
  - issues with legacy code may occur
  - compiler automatically inserts casts etc.
  - That's the reason why this doesn't work:
    - new E() //error
- Generic methods
  - class Collections:

```
/** Replace all occurrences in a list. */
public static <T> boolean replaceAll (List<T> list,
                                     T oldVal, T newVal)
```
  - a method can have its own type parameter

## Generic Types: Comparable

```
/** An object that can be compared. The interface imposes a total
    ordering. T is the type of objects that this object may be
    compared to. */
```

```
public interface Comparable< T > {
    /**Compare this object to o. Return -1 if this object is smaller,
        0 if they are equal, and 1 if this object is greater. */
    int compareTo(T o);
}
```

- Needed for sorting and searching objects
  - implemented by String, Integer, Double, many others
- T is usually the same type
  - example:
    - class String implements Comparable< String >