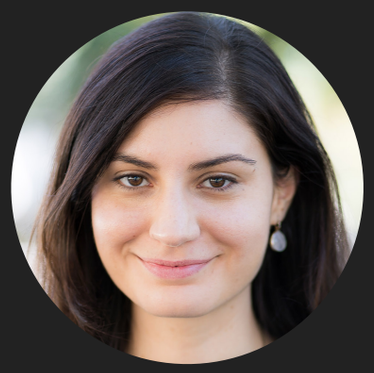
FUNDAMENTALS

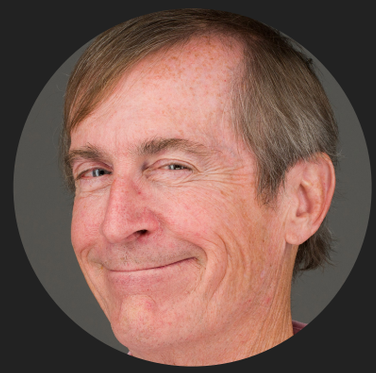
CS062

DATA STRUCTURES AND ADVANCED PROGRAMMING

5: Generics, Packages, and JavaDoc



Alexandra Papoutsaki
LECTURES



Mark Kampe
LABS

Lecture 5: Generics, Packages, JavaDoc

- Generics
- Enum
- Packages
- JavaDoc

Some slides adopted from Algorithms, 4th Edition and Oracle tutorials

Generics

- Compile-time errors can be easier to fix than run-time errors.
- Java introduced generics (similar to templates in C++) to help move more bugs to compile-time (easier to debug!), eliminate casting, and improve abstraction. E.g.,

```
List list = new ArrayList();
list.add("hello");
String s = (String) list.get(0);
```

Is now:

```
List<String> list = new ArrayList<String>();
list.add("hello");
String s = list.get(0);   // no cast
```

- Generics enable types (that is classes and interfaces) to be used as parameters when defining classes, interfaces, and methods.

Formal and actual type parameters

```
public interface List <E> {
    void add(E x);
    Iterator<E> iterator();
}
```

Formal type parameters

```
public interface Iterator<E> {
    E next();
    boolean hasNext();
}
```

- In the invocation (e.g., `List<Integer>`) all occurrences of the formal type parameters are replaced by the actual type argument (e.g., `Integer`).

Generic classes

```
class name <T1, T2, …, Tn> {…}
```

- A type variable can be any non-primitive type (class, interface, array)
- E: element (common in data structures), T: type, K: key, V: value, N: number, etc.

```
/**
* Generic version of the Box class.
* https://docs.oracle.com/javase/tutorial/java/generics/types.html
* @param <T> the type of the value being boxed
*/

public class Box<T> {
    private T t;

    public void set(T t) { this.t = t; }
    public T get() { return t; }
}
```

- Invocation: `Box<Integer> integerBox = new Box<Integer>();`

Multiple Type Parameters Example

```
public interface Pair<K, V> {
    public K getKey();
    public V getValue();
}

public class OrderedPair<K, V> implements Pair<K, V> {
    private K key;
    private V value;

    public OrderedPair(K key, V value) {
        this.key = key;
        this.value = value;
    }
}
Pair<String, Integer> p1 = new OrderedPair<String, Integer>("Even", 8);

OrderedPair<String, Box<Integer>> p = new OrderedPair<String, Box<Integer>>("primes", new
Box<Integer>(...));
```

Generic methods

```
modifier (static) <T1, T2, …, Tn> return-type name(list of type parameters){…}}
```

- The type parameter's scope is limited to the method which is declared.
- Static, non-static generic methods, generic class constructors are allowed.
- Type inference: allows you to invoke a generic method as an ordinary method, without specifying a type between angle brackets.
- E.g., `className/objectName.genericMethod(arguments);`

Practice Time

- Write a generic method to exchange the positions of two different elements in an array.
  - You are given the array and both indices that need to be swapped.

Answer

```
public static <T> void swap(T[] a, int i, int j) {
    T temp = a[i];
    a[i] = a[j];
    a[j] = temp;
}
```

Lecture 5: Generics, Packages, JavaDoc

- Generics
- Enums
- Packages
- JavaDoc

The Enum type

- Special data type that enables for a variable to be a set of predefined constants.
- The variable needs to be equal to one of the predefined values of the set.
- You should use enum types any time you need to represent a fixed set of constants, e.g., days, months, compass directions, etc.

- Enum's type fields are in ALL CAPS.

```
public enum Day {
    SUNDAY, MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY
}
```

- Then use as an object anywhere.
  - e.g., `Day.values();`

Lecture 4: Generics, Packages, JavaDoc

- Generics
- Enums
- Packages
- JavaDoc

What is a package?

- A grouping of related classes and interfaces that provides access protection and name space management.
- e.g., `java.lang` for fundamental classes or `java.io` for classes related to reading input and writing output.
- Packages correspond to folders/directories.
- Lower-case names.
- `package whatevername;` at top of file.
- `import graphics.*;` for including all classes/interfaces.
- or `import graphics.Circle;` for more specific access.

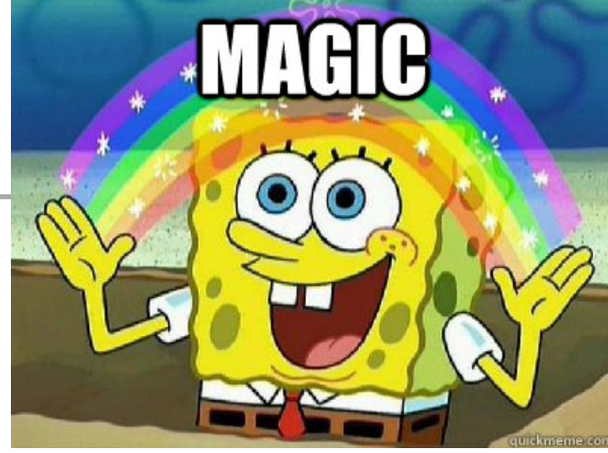
https://docs.oracle.com/javase/tutorial/java/package/packages.html

Access modifiers

Modifier	Class	Package	Subclass	World
`public`	Y	Y	Y	Y
`protected`	Y	Y	Y	N
No modifier	Y	Y	N	N
`private`	Y	N	N	N

Lecture 5: Casting, Generics, Packages

- Casting
- Generics
- Enums
- Packages
- JavaDoc



Java Documentation Generation System

- Reads JavaDoc comments and gives HTML pages
- JavaDoc comment = description written in HTML + tags
- Enclosed in `/**  */`
- Must precede class, variable, constructor or method declaration
- For class:
  - **@author** author name – classes and interfaces
  - **@version** date - classes and interfaces
- For method:
  - **@param** param name and description – methods and constructors
  - **@return** value returned, if any – methods
  - **@throws** description of any exceptions thrown - methods

https://www.oracle.com/technetwork/articles/java/index-137868.html

Lecture 5: Generics, Packages, JavaDoc

- Generics
- Enums
- Packages
- JavaDoc

Readings:

- Oracle's guides:
  - Generics: https://docs.oracle.com/javase/tutorial/java/generics/index.html https://docs.oracle.com/javase/tutorial/extra/generics/intro.html
  - JavaDoc: https://www.oracle.com/technetwork/articles/java/index-137868.html
- Textbook:
  - Page 122
- Textbook Website:
  - Generics: https://algs4.cs.princeton.edu/13stacks/