Objective

Bonus Slides

- Generic Programming
- Builders

Reference

Generic Programming : Chapter 18.1 to 18.4

Lamda: tutoirals

https://medium.freecodecamp.org/learn-these-4-things-and-working-with-lambda-expressions-b0ab36e0fffc

http://tutorials.jenkov.com/java/lambda-expressions.html

Builders: tutorial

http://www.vogella.com/tutorials/DesignPatternBuilder/article.html

Stream: Chapter 19

Tutorial:

https://winterbe.com/posts/2014/07/31/java8-stream-tutorial-examples/

Generic Classes

Generic programming is a technique that can be used by different types.

ArrayList uses generic programming technique.

```
ArrayList <BankAccount> list;
```

Generic class parameters are inclosed in <>

ArrayList<E>, where E is the type variables

Example:

```
ArrayList <BankAccount> list;
list = new ArrayList<BankAccount>();
```

Generic Classes

When you instantiate a generic class, the type that you supply replaces all occurrences of the type variable in declaration.

Example:

```
public void add(E element)
public E get(int index)
```

are replaced by

```
public void add (BankAccount element)
public BankAccount get(int index)
```

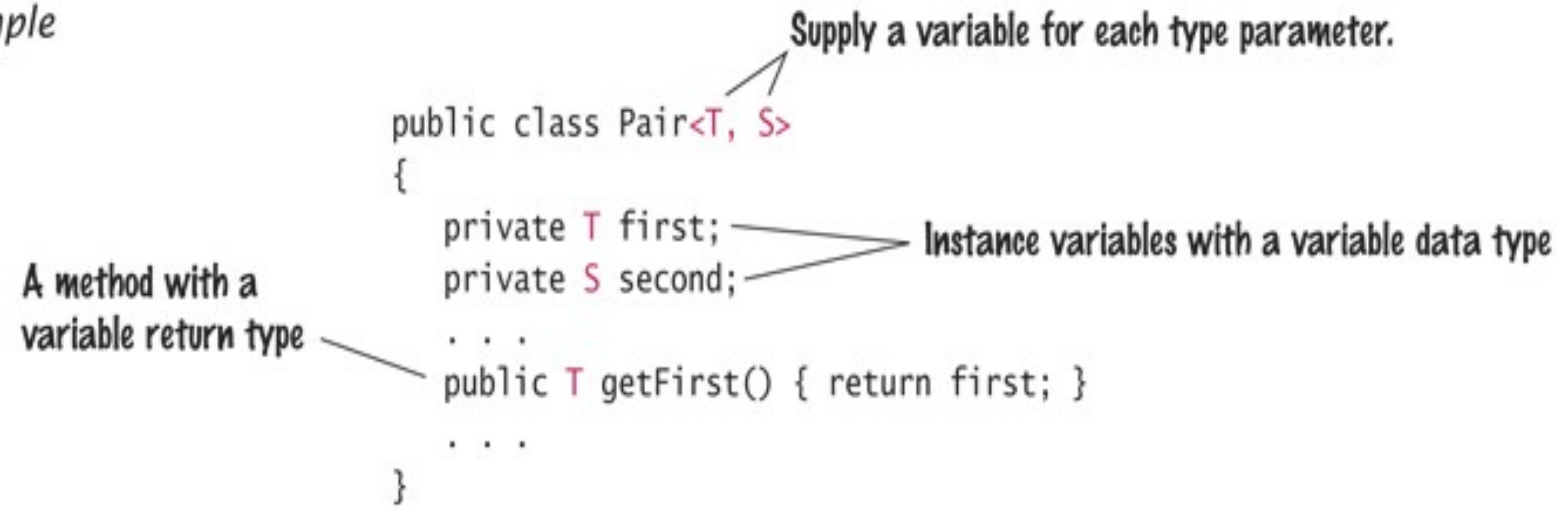
Note that you can achieve the same result by using Objects, but generic classes are safer.

Implementing Generic Classes

Syntax

```
accessSpecifier class GenericClassName<TypeVariable1, TypeVariable2, . . .>
{
   instance variables
   constructors
   methods
}
```

Example



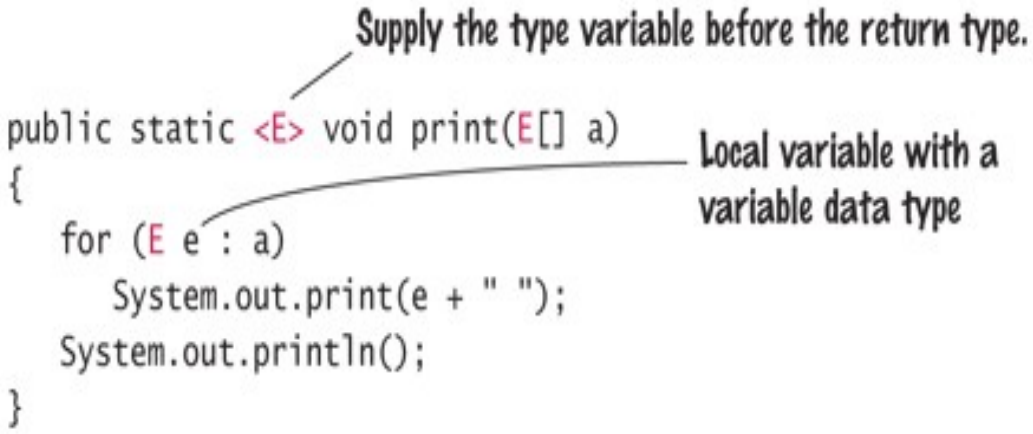
Check sample1

Implementing Generic Method

Syntax

modifiers <*TypeVariable*$_1$, *TypeVariable*$_2$, . . .> *returnType methodName*(*parameters*)
{
   body
}

Example




```
public static <E> void print(E[] a)
{
   for (E e : a)
      System.out.print(e + " ");
   System.out.println();
}
```

Check sample2, sample3, sample4, and sample5

Standard Variable Types

Type Variable	Meaning
E	Element type in a collection
K	Key type in a map
V	Value type in a map
T	General type
S, U	Additional general type

Builder Utility

```
public class Address{
   private int number;
   private String street;
   private String city;
   private String province;
   private String zipCode;
    ...
}
```

Builder Utility

```
To create an object of class Address:
// call constructor
Address address1 = new Address(100, "W 49th
Ave", "Vancouver", "BC"," V5Y2Z6");

Or use Builder utility
Address ad2 = new AddressBuilder().
                  setNumber(100).
                  setStreet("W 49th Ave").
                  setCity("Vancouver").
                  setProvince("BC").
                  setZipcode("V5Y2Z6").
                  build();
```

Check e1, e2, and e3

Streams

Java 8 Streams are not I/O streams.

It is chained operations, like:

```
Stream.of("a1", "a3", "a1")
    .map(s -> s.substring(1))
    .mapToInt(Integer::parseInt)
    .max()
    .ifPresent(System.out::println);
```

Output: 3

Suggested Tutorial:

https://winterbe.com/posts/2014/07/31/java8-stream-tutorial-examples/