**CSCI 2302**

Generics Chapter

Using Generic Methods Lab

Contents

[Intro 1](#_Toc86236540)

[Notes 1](#_Toc86236541)

[Advantages 1](#_Toc86236542)

[Learning Goals 2](#_Toc86236543)

[Task 2](#_Toc86236544)

[Submit 4](#_Toc86236545)

Intro: Wouldn’t it great if we could implement/define a method that was not restricted by a data type, but instantiated with a specific data type? Yes, it would – and Generics allow just that.

Notes: Let’s recap about Java. Java is strongly data typed. This means that in order to have a variable it must declared with a data type and once that data type is declared, it cannot be changed. Generics allow a class to be used by several data types; this is just one of the advantages of generics. Generics have several advantages:

Advantages**:**

* Generics enable you to detect errors at compile time rather than at runtime.
* Generics enable you to reuse code independent of the data types, write a method/class/interface one time and have it applicable to any data types needed.
* Generics eliminate the need to do individual type casting.

Generics are defined with <>. The data type that is specified within the <> is the formal generic type. What the formal generic type is replaced with is the actual concrete type. This replacing is called generic instantiation.

By convention, a single capital letter usually the letter E or the letter T.

By convention, type parameter names are single, uppercase letters. This stands in sharp contrast to the variable naming conventions that you already know about, and with good reason: Without this convention, it would be difficult to tell the difference between a type variable and an ordinary class or interface name.

The most commonly used type parameter names are:

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

K - Key

N - Number

T - Type

V - Value

S,U,V etc. - 2nd, 3rd, 4th types

To declare a generic method, you place the generic type <E> immediately after the keyword static in the method header.



There are 2 ways to invoke the generic method:

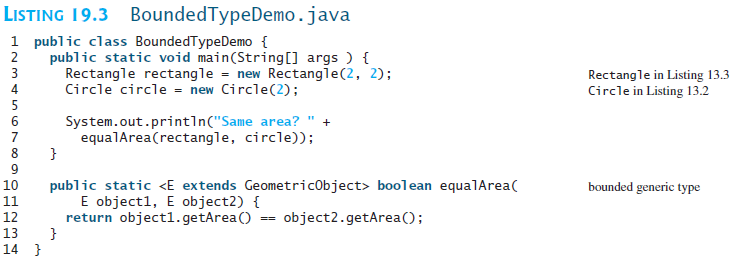
1. Simply invoke the method with the method name and the actual parameter, example:



1. State the class name.<actualType>methodName(actual parameter); example:



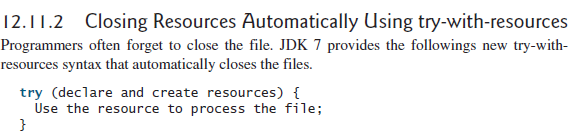
Example:



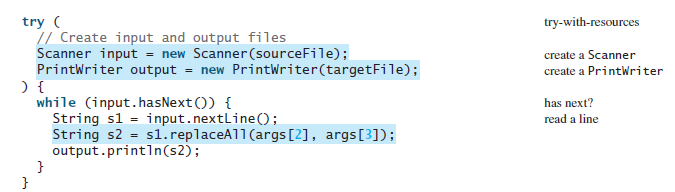
Learning Goals**:** To define and use a generic method.

Task**:** Complete the steps outlined below

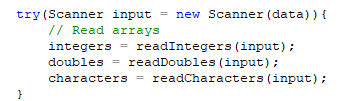
1. Download GenericsMethodsLab.javaand rename the file to mysfasuserName\_GenericsMethodsLab.java.
2. Add a try with resources to set up a scanner to read from the file, make sure to set this up so that the resource is closed after the resource is needed (Section 11.2 in the Exception Handling and Text I/O chapter).



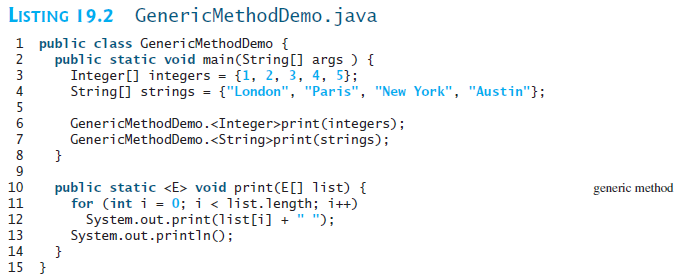
Example: (***pay******attention*** to the () and the {} )



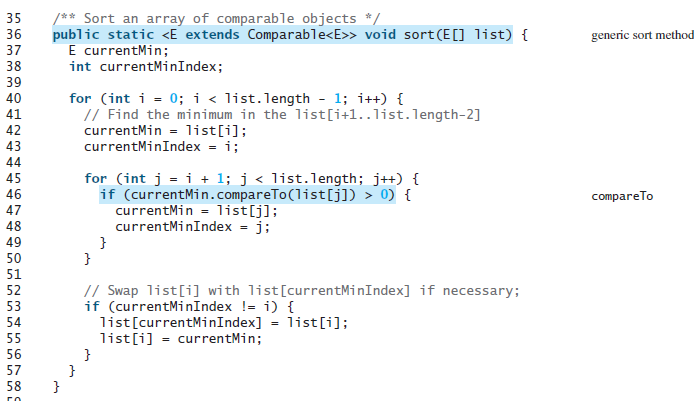
For this code, all you need to do is add the try() and put the {} in the correct spot; the “reading” in methods are already stated.



1. Add the code to the print method (GenericMethodDemo.java).



1. Add a generic sort to the sort method (GenericSort.java).



Input**:** Read the input from a file, genericsIn.txt. (format: first number informs you of each array size; integers array data, doubles array data, characters array data).

Data**:** 3 3 9 4

3 50.3 23.8 12.4

3 Q W E

Sample run**:**

The integer array: 3 9 4

The double array: 50.3 23.8 12.4

The character array: Q W E

The Sorted Arrays are:

3 4 9

12.4 23.8 50.3

E Q W

Submit: mysfasuserName\_GenericsMethodsLab.java file in the Dropbox in in Brightspace by D2L.