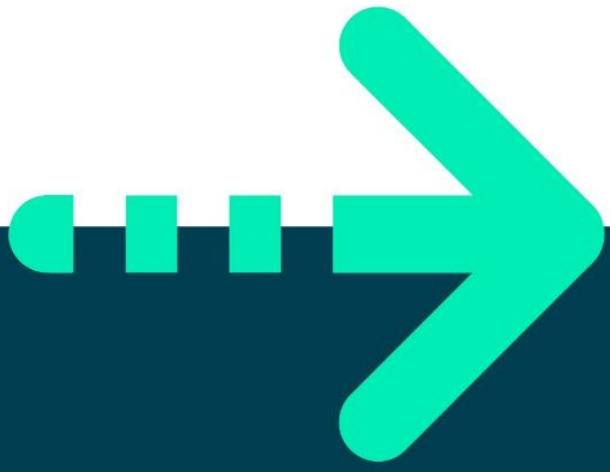
EXERCISE 7, GENERIC INTERFACES



Exercise 7, Generic interfaces

Objective

To be able to use and understand the role of generic interfaces.

Overview

Initially you will be working in a project which declares, creates and manipulates a list of 'person' references. You will use generic collection class `Arrayist<E>` and a generic interface `Comparable<E>` to sort the entries in the List.

In Part two you will take your work to date and enhance it using another interface `Comparer<E>`, which will enable you to sort your List in an alternative sequence.

Part 1 – Using a Generic class and Comparer<E>

Step by step

Creating, filling and printing a list of person references

1. Open the Labs project
2. Add a new package called lab07
3. Add a new class called Program with a main() method
4. Add a new class called Person
   a. Add a following fields:

      String name;
      int age;

   b. Create a constructor to set the name and age of a person
5. Override the `toString() method` that prints out a padded `name` and `age` tab-separated.
6. Open the main() method and create an `ArrayList<Person>` called people.
7. Add 6 Persons to the people ArrayList given the following ages and names:

```
int[] ages = { 36, 53, 51, 21, 41, 19 };

String[] names = {
 "Holland", "Turner", "Powell", "Vicious", "Lyndon", "Spears"
};
```

   Tip: use a normal for loop using index of 0 to names.size()

Implementing Comparable<Person>

8. If you invoke Collections.*sort*(*people*) method in the main() method, you'll get the following exception:

   ```
   Solution.GenericMethods.Person cannot be cast to
   java.lang.Comparable
   ```

   You get the above error because the system doesn't know how to compare one element of the people with another. To tell the system how to compare two elements, Person class has to implement the Comparable<Person>

9. Open the Person class and implement the Comparable<Person> to sort people by age.

   Tip: Please see the slides if you're not sure how to do this.

10. Back in the main() method, type Collections.*sort*(*people*); to sort people by age.

11. Write an enhanced for loop to display every person's age and name in the people list.

    Tip: better create a method to display people.

Sorting by alternative criteria

Part 2 – Sorting the list – Implementing Comparable<Person>

Now it's time to sort the `ArrayList<Person>` by `N`ame without changing its code.

1. Let's create another class that enable us to sort people by name.

   a. Add a class called PersonNameComparer

   b. This class must implement Comparator<Person>

   Tip: use the String's compare() method to compare names.

2. Back in the *main()* method, call the *Collection.sort()* method like: Collection.sort(people, new *PersonNameComparer()*);

3. Display people to make sure names are sorted.

** End **