**44-542 Object-Oriented Programming**

**Sorting with Lambda Expressions**

1. Create a project with a package named **dogs**.
2. Add the **Dog** class shown here to your project.

**package dogs;**

**public class Dog {**

**private String name;**

**private String breed;**

**private int age;**

**private int weight;**

**public Dog(String name, String breed, int age, int weight) {**

**this.name = name;**

**this.breed = breed;**

**this.age = age;**

**this.weight = weight;**

**}**

**public String getName() {**

**return name;**

**}**

**public String getBreed() {**

**return breed;**

**}**

**public int getAge() {**

**return age;**

**}**

**public int getWeight() {**

**return weight;**

**}**

**@Override**

**public String toString() {**

**return String.format(**

**"%-10s %-10s %3d %3d", name, breed, age, weight);**

**}**

**}**

1. Create a main class with name **Sorting**. All of the remaining instructions describe what to include in method **main**.
2. We will create some **Dog** objects and sort them later, but let's cover a few basics using a list of numbers.
3. In method main, create a list of integers named nums and containing the values 2, 9, 4, 6, 1, 8, 5. Here is an easy way to do this:

**Integer[] a = {2, 9, 4, 6, 1, 8, 5};**

**List<Integer> nums = Arrays.asList(a);**

OR

**List<Integer> nums = Arrays.asList(2, 9, 4, 6, 1, 8, 5);**

If you want to know more, look up method **asList** in the **Arrays** class.

1. Print the elements in **nums**, with a heading that says "Integers in original order".
2. Java 1.8 introduced a default method named **sort** in the **List** interface. It has one argument, which is a comparator. The elements in the **List** object that invokes **sort** are sorted according to the comparator passed in the parameter. The general syntax we will use here is

**nums.sort(*some comparator*)**

1. There are several different ways to supply the comparator. We can use a lambda expression, or we can invoke a method that returns a comparator. Here is an example using a lambda expression:

**nums.sort((x,y) -> Integer.compare(x, y));**

This describes how two elements in the list should be compared – just use the compare method from the Integer class. This method compares two integers numerically, in the usual way.

1. An alternative way to write this is shown below. We can leave out the word Integer, and Java will figure out what we mean (there is a lot more "inference" in Java 8 than in some earlier versions).

**nums.sort((x,y) -> compare(x, y));**

1. There is yet another way to write the **sort** statement, using the method reference operator **::**.

**nums.sort(Integer::compare);**

This statement tells Java to use the **compare** method in the **Integer** class to compare integers.

1. Select one of the methods described above to sort the integers in **nums**, and then print the list with the heading "Integers in increasing order".
2. Suppose we want to sort **nums** in reverse order. The **Collections** class has a static method named **reverseOrder** that will do exactly that. We can call the **sort** method as we did above, and then invoke the **reverseOrder** method to produce a comparator that can be used as the argument for the **sort** method:

**nums.sort(Collections.reverseOrder());**

1. Using the **reverseOrder** method as described above, sort **nums** in reverse order. Then print **nums** again, with the heading "Integers in decreasing order".
2. Now let's sort some dogs. Begin by creating an **ArrayList** of **Dog** objects using the code shown below:

**List<Dog> dogList = new ArrayList();**

**dogList.add(new Dog("Carmen", "Poodle", 11, 48));**

**dogList.add(new Dog("Leonard", "Poodle", 6, 46));**

**dogList.add(new Dog("Lucy", "Beagle", 3, 19));**

**dogList.add(new Dog("Sally", "Saluki", 2, 54));**

**dogList.add(new Dog("Zelda", "Poodle", 14, 43));**

**dogList.add(new Dog("Sam", "Saluki", 8, 56));**

**dogList.add(new Dog("Flash", "Greyhound", 7, 62));**

**dogList.add(new Dog("Eve", "Poodle", 9, 51));**

**dogList.add(new Dog("Ben", "Beagle", 5, 22));**

**dogList.add(new Dog("Ralph", "Saluki", 6, 49));**

1. Print the list, with each **Dog** on a different line, and using the **toString** method of the **Dog** class. The heading should be "Dogs in original order."
2. Sort the list in alphabetical order by name. Here are two ways to do that:

**Collections.sort(**

**dogList,**

**(d1, d2) -> d1.getName().compareTo(d2.getName()));**

OR

**dogList.sort((d1,d2) -> d1.getName().compareTo(d2.getName()));**

1. Print the list with heading "Dogs in alphabetical order by name".
2. Sort the list in increasing order by age, using lambda expressions as in the previous examples. (See if you can figure out how to do this.)
3. Print the list with heading "Dogs in increasing order by age".
4. Sort the list in decreasing order by weight, using lambda expressions as in the previous examples. (See if you can figure out how to do this.)
5. Print the list with heading "Dogs in decreasing order by weight".

Output:

**Integers in original order: 2 9 4 6 1 8 5**

**Integers in increasing order: 1 2 4 5 6 8 9**

**Integers in deccreasing order: 9 8 6 5 4 2 1**

**Dogs in original order:**

**Carmen Poodle 11 48**

**Leonard Poodle 6 46**

**Lucy Beagle 3 19**

**Sally Saluki 2 54**

**Zelda Poodle 14 43**

**Sam Saluki 8 56**

**Flash Greyhound 7 62**

**Eve Poodle 9 51**

**Ben Beagle 5 22**

**Ralph Saluki 6 49**

**Dogs in alphabetical order by name:**

**Ben Beagle 5 22**

**Carmen Poodle 11 48**

**Eve Poodle 9 51**

**Flash Greyhound 7 62**

**Leonard Poodle 6 46**

**Lucy Beagle 3 19**

**Ralph Saluki 6 49**

**Sally Saluki 2 54**

**Sam Saluki 8 56**

**Zelda Poodle 14 43**

**Dogs in increasing order by age:**

**Sally Saluki 2 54**

**Lucy Beagle 3 19**

**Ben Beagle 5 22**

**Leonard Poodle 6 46**

**Ralph Saluki 6 49**

**Flash Greyhound 7 62**

**Sam Saluki 8 56**

**Eve Poodle 9 51**

**Carmen Poodle 11 48**

**Zelda Poodle 14 43**

**Dogs in decreasing order by weight:**

**Flash Greyhound 7 62**

**Sam Saluki 8 56**

**Sally Saluki 2 54**

**Eve Poodle 9 51**

**Ralph Saluki 6 49**

**Carmen Poodle 11 48**

**Leonard Poodle 6 46**

**Zelda Poodle 14 43**

**Ben Beagle 5 22**

**Lucy Beagle 3 19**