

SWE5202 Data Structures and Algorithms

Practical Exercises – Arrays

Exercise 1

Create a new Java project (call it Array) and in this project create a new class called Dog which implements the Comparable interface and has the attributes and methods shown below.

|  |  |
| --- | --- |
| Class name: Dog (Attributes) |  |
| private Long chipID | A unique number stored on a chip and inserted into the dog |
| private String name | The dog’s name e.g. Fido |

|  |  |
| --- | --- |
| Class name: Dog (Methods) | |
| public Dog(Long chipID, String name) | ctor – creates a Dog object using data passed by the parameters. |
| public String toString() | returns a string e.g. ‘Fido (123456)’ |
| public int compareTo(Object obj) | returns -1, 0 or +1 depending on comparison with the object passed |
| In addition provide suitable getters and setters for the class attributes | |

You have probably not used the toString and compareTo methods before so here is the code for these 2 methods.

public String toString(){ String s;

s= name + " (" + chipID + ")";

return s;

}

public int compareTo(Object obj) {

Dog d = (Dog) obj; if(d == null) return 1;

else

return chipID.compareTo(d.chipID); }

BEFORE continuing ensure that your code is syntactically correct – if you have any errors or warnings that you cannot resolve then ask!

The toString and compareTo methods are ‘special’ compared with the others we have created and this can be seen in the Eclipse IDE.

Page 1 of 4 04/10/23 A. Razak How does Eclipse indicate they are special?

Why are they special?

In your project create a class called Test001 and inside the class enter the following code.

public static void main(String[] args) {

Dog d1 = new Dog(new Long(56498), "Fido");

Dog d2 = new Dog(new Long(34999), "Rover");

System.out.println(d1);

System.out.println(d2);

}

# Experiment 1

Run the program and notice what is displayed in the output window? Compare the output to the Dog class, are there any conclusions that can be made? Not sure, then comment out the toString method and run the program again any difference?

Answer

*The toString method helps print the output stored in that specific memory space but if toString method is commented then the program prints the memory space instead.*

# Experiment 2

Modify the method so that it looks like this

public static void main(String[] args) {

Dog d1 = new Dog(new Long(56498), "Fido");

Dog d2 = new Dog(new Long(34999), "Rover");

System.out.println(d1);

System.out.println(d2);

if(d1.compareTo(d2) == 0)

System.out.println("Same doggy"); else

System.out.println("Different doggy"); }

Change the name of d2 to Fido and run the program again. Do the dogs match?

*No the dogs do not match.*

# Experiment 3

Change the name of d2 back to Rover and its chip ID to 56498 and run the program again. Do the dogs match?

ANSWER

*Yes the output is same doggy*

# Experiment 4

Change the Dog class so that dogs with the same name match.

Describe in your own words the rules for implementing the compareTo method.

*The comparison between the values needs to be of the same data type.*

Why is the toString method useful?

*The information stored is easily readable for humans. It shows what exactly is stored in a particular memory space.*

Include the code for the compareTo method you created in experiment 4.

**public** **int** compareTo(Dog otherDog) {

**if** (**this**.chipID.equals(otherDog.getChipID())) {

// If chipIDs are equal, compare based on name

**return** **this**.name.compareTo(otherDog.getName());

} **else** {

// Compare based on chipID

**return** **this**.chipID.compareTo(otherDog.getChipID());

}

}

What is the difference between Long and long and what similar cases can you find?

*long can not be null whereas Long can be null. (differences)*

*Long is an object whereas long is a primitive data type that can be found in the java library.(differences)*

*Both Long and long store integer values.(similarities)*

*Long occupies more space than long because its an object.(differences)*

# Exercise 2

In your project create a class called Test002 and inside this class enter the following code

private static final int NO\_DOGS\_IN\_POUND = 200000; private static final int NO\_REPORTED\_LOST = 2000; private static final int RANGE = NO\_DOGS\_IN\_POUND \* 2; private static final Random rnd = new Random();

private Dog dogPound[] = new Dog [NO\_DOGS\_IN\_POUND]; private Dog reportedLost[] = new Dog [NO\_REPORTED\_LOST];

public Test002(){ long id;

for(int i = 0; i < NO\_DOGS\_IN\_POUND; i++){ id = Math.abs(rnd.nextInt()) % RANGE;

dogPound[i] = new Dog(id, "");

}

for(int i = 0; i < NO\_REPORTED\_LOST; i++){ id = Math.abs(rnd.nextInt()) % RANGE;

reportedLost[i] = new Dog(id, "");

}

}

public void pause(long delay){

try{

Thread.sleep(delay);

}

catch (Exception e){

e.printStackTrace();

}

}

// Perform single sequential search

public boolean search(Dog value){

int i = 0;

while(i < NO\_DOGS\_IN\_POUND && dogPound[i].compareTo(value) != 0){

i++;

}

return (i < NO\_DOGS\_IN\_POUND);

}

public void performSequentialSearch(){

long time;

int nbrFound = 0;

System.out.print("#### SEQUENTIAL SEARCH (Data Set Size ");

System.out.println(NO\_DOGS\_IN\_POUND + " ) ###");

time = System.currentTimeMillis(); for(int i = 0; i < NO\_REPORTED\_LOST ; i++){

if(search(reportedLost[i]))

nbrFound++;

}

time = System.currentTimeMillis() - time;

System.out.print(" Number matched : " + nbrFound);

System.out.println(" out of " + NO\_REPORTED\_LOST + " searchs");

System.out.println(" Time taken : " + time + " milliseconds\n");

}

public void performBinarySearch(){

long time;

int nbrFound = 0;

System.out.print("#### BINARY SEARCH (Data Set Size ");

System.out.println(NO\_DOGS\_IN\_POUND + " ) ###");

time = System.currentTimeMillis(); for(int i = 0; i < NO\_REPORTED\_LOST ; i++){

if(Arrays.binarySearch(dogPound, reportedLost[i]) >= 0)

nbrFound++;

}

time = System.currentTimeMillis() - time;

System.out.print(" Number matched : " + nbrFound);

System.out.println(" out of " + NO\_REPORTED\_LOST + " searchs");

System.out.println(" Time taken : " + time + " milliseconds\n");

}

public void execute(){

long time;

performSequentialSearch();

pause(100);

// Sort the data set array int nbrFound = 0;

System.out.print("SORTING " + NO\_DOGS\_IN\_POUND + " elements took "); time = System.currentTimeMillis(); Arrays.sort(dogPound);

time = System.currentTimeMillis() - time; System.out.println("" + time + " milliseconds\n"); pause(100);

pause(100);

performBinarySearch();

} public static void main(String[] args) { Test002 that = new Test002(); that.execute();

}

Once you have entered this program run it and study the output. Comment out the line in yellow and run the program again study the output.

What conclusions can we draw from these two sets of results?

*The outcome is that arranging the information with the dogPound array produces more effective results to the search. When the information is mixed the search becomes complicated and less efficient which results in longer execution time and less matches found.*