**Assignment 12 – Functional Programming and Lambda Expressions**

**CSC 300 Fall 2024**

**Total Points – 161**

Notes: All homework must be submitted via e-mail. All parts of assignment must be submitted in a single e-mail with multiple attachments when required.

Notes: All homework must be submitted via e-mail. All parts of assignment must be submitted in a single e-mail with multiple attachments. E-mail address is:

[csc300csudh@gmail.com](mailto:csc300csudh@gmail.com)

Each program/class is to be submitted in a separate file with the file name being the class name with extension .java. I only need the source file. (Only relevant when software is assigned).

The other homework file should be submitted with your name using the following format:

**RosenthalH\_AZ.docx**, where Z is the assignment number. Note: use your name, not mine

1. **Short Answers (2 points each)**
2. Lambdas allow us to treat \_\_\_\_A\_\_\_\_
3. Code as data
4. Data as code
5. Code as classes
6. All of the above
7. Lambda expressions are based on \_\_\_\_B\_\_\_\_
8. Procedural programming
9. Functional programming
10. Data programming
11. All of the above
12. In Java Interfaces, methods can be: \_\_\_\_D\_\_\_\_
13. default
14. abstract
15. static
16. All of the above
17. Which of the following is a valid java expression? \_\_\_\_C\_\_\_\_
18. String a, String b -> System.out.print(a+ b);
19. () -> return;
20. (int i) -> i;
21. (int i) -> i++; return i;
22. Given the snippet below which is a valid statement? \_\_\_\_C\_\_\_\_

Interface myIF {int aMethod(String s);}

1. myIF a = a -> a.length();
2. myIF x = y -> {return y;};
3. myIF b = (String s) -> 1;
4. myIF s = "2" -> Integer.parseInt(s);
5. A lambda expression can be used as (Hint: List 2 letters for the 2 correct answers \_\_\_A,C\_\_\_\_\_
6. As a method argument
7. As a conditional expression in an if statement
8. In a return statement
9. In a throw statement
10. Given the following lambda snippet which interface can provide the proper functional interface for this lambda expression? \_\_B\_\_\_\_\_\_

() -> 7 \* 12.0;

interface A {

default double m() {

return 4.5;

}

interface B {

double m();

}

interface C {

int m();

}

interface D {

double m(Integer... i);

}

1. Based on the Java syntax rules, which of the following are not valid lambda expressions? (mark with V or NV)
2. ()->{} \_\_\_\_V\_\_\_\_
3. () -> "Raoul" \_\_\_\_NV\_\_\_\_
4. () -> { return "Mario"; }\_\_\_V\_\_\_\_\_
5. (Integer i) -> return "Alan" + i; \_\_\_\_NV\_\_\_\_
6. (String s) -> { "Iron Man"; } \_\_\_NV\_\_\_\_\_
7. **Fill in the Blanks (2 points each)**
8. Lambda expressions implement **\_\_\_\_\_\_\_functional interface\_\_\_\_\_\_\_**
9. A(n) **\_\_\_\_\_\_\_\_\_lambda expression\_\_\_\_\_\_\_\_\_\_** represents an anonymous method—a shorthand notation for implementing a functional interface.
10. **General Questions (4 points each)**
11. Write a no-argument lambda expression that implicitly returns the String "I am a Java Programmer”

ANS: () -> “I am a Java Programmer”

1. Create a two-parameter lambda expression with types double, double as the inputs that returns the value of the first parameter raised to the power of the second parameter.

ANS: (double a, double b) -> {return Math.pow(a, b)}

**Programs**

1. **Simple Lambda Program (25)**

Given the following functional interface:

interface **TwiceSquareRoot**

{

double twiceSquareRoot (double b);

}

Write a simple class **LambdaTwiceSquareRoot** with a main method that implements the interface so that it returns twice the square root of the input value.

The interface should be implemented two ways. e1 creates a function without a return statement. e2 creates a function with a return statement inside the lambda expression. For each case read in the inputs.

Write simple output statements to demonstrate the two functions.

Sample input/output:

Please enter the value as a type double: 9.0

Without return double the square root is 6.000

Please enter the values as a type double: 27.0

With return double the square root is 10.392

1. **GenIFTester (30)**

Create two files for this exercise, one for the functional interface GenericInterface and one to test it called **GenIFTester**.

**GenericInterface**

This is a generic class of type T.

It has one abstract method called func that has a parameter of type T and returns a value of type T.

**GenIFTester**

This program can all be done in main

Create a Scanner to read from the keyboard.

Create a lambda function called reverse that implements GenericInterface

This lambda accepts a variable of type String and reverses all the characters of the parameter. It returns the reversed String.

Then do three times

Request an input String.

Use reverse to reverse the String and print out the original String and the reversed String (see Sample IO below).

Create a lambda function called factorial that implements GenericInterface

This lambda accepts a variable of type int and calculates its factorial value. It returns the int value. (Note: Boxing will take place automatically or you can use Integer in place of int if you like

Then do three times

Request an input integer.

Use factorial to calculate the factorial value and print out the original number and factorial (see Sample IO below).

**Sample Input/Output**

Enter a string to be reversed:

abcde

The entry abcde reversed is edcba

Enter a string to be reversed:

My Day Is Great 100%

The entry My Day Is Great 100% reversed is %001 taerG sI yaD yM

Enter a string to be reversed:

39-87-Grand

The entry 39-87-Grand reversed is dnarG-78-93

Enter an integer to be factorialised:

4

factorial of 4 = 24

Enter an integer to be factorialised:

6

factorial of 6 = 720

Enter an integer to be factorialised:

9

factorial of 9 = 362,880

1. **PlayfulPets (70)**

**Pet** class

This class has a static List variable of type Pet named pets that is instantiated as an ArrayList.

This class has the following private instance variables:

String name

String animal

String breed

String color

double price

Each instance variable has a simple mutator and accessor.

There is a default constructor that does nothing.

There is a second constructor with parameters for each of the five instance variables.

It calls the mutator for each parameter in order to set the corresponding instance variable.

There is a toString method that prints out a line like this:

Max: a black poodle dog purchased for $540.50

There is an equals method that returns true if the item being compared with is also a Pet and if the animal and breed are the same.

**PetMatcher** interface

PetMatcher is a functional interface with both an abstract and default method.

method **matchPet** is the abstract method (don’t use the word abstract since we are in an interface.) This method returns a List of type Pet and has a Pet as an input parameter

default method **firstPet** has an input of type Pet and a return value of type Pet.

Get the index of the first matching Pet in Pet.pets (the ArrayList in Pet) (use indexOf)

if the index is less than 0 (indexOf does this for you)

return null

else

return a reference to the Pet at that index

**PlayfulPets** class

This class has a **main** method and one additional method called **matchPetsFromTheList**. It also creates two different implementations of the PetMatcher’s functional interface’s abstract method.

method **main**

Create five Pets and add them to the Pet.pets (the static ArrayList of Pet)

You can use the following:

“Scruffy","dog","poodle","brown",895.00

“Meow","cat","siamese","white",740.25

“Max","dog","poodle","black",540.50

“Cuddles","dog","pug","black",1282.75

"HotDog","dog","dachsund","brown",621.75

“Slider","snake","garden","brown",320.00

Please use a Lambda Expression for breedMatcher

Create a new PetMatcher called **breedMatcher** that implements petMatch returning a List of all those Pets of the same breed. In this case we don’t override the firstPet method.

The following instances of PetMatcher (priceMatcher and color Matcher) should be done using inner classes. This is necessary in order to override the default method in the easiest possible way.

Create a new PetMatcher called **priceMatcher** that implements petMatch returning a List of all those Pets less than or equal to the price of the input parameter Pet. priceMatcher also overrides the firstPet method, creating a new firstPet that searches Pets.pet for a Pet less than or equal to the price of the input Pet, returning a reference to the first such instance of such a Pet. If no such Pet exists return null.

Create a new PetMatcher called **colorMatcher** that implements petMatch returning a List of all those Pets equal to the color of the input parameter Pet.(Remember color is a String so use the equals method on the colors.) colorMatcher also overrides the firstPet method, creating a new firstPet that searches Pets.pet for a Pet with a color that equals the color of the input Pet, returning a reference to the first such instance of such a Pet. If no such Pet exists return null.

Make the following calls in main:

matchPetsFromTheList("Pugs",breedMatcher, new Pet(null, "dog", "pug", null, 0.0));

matchPetsFromTheList("Pets for $650 or less",priceMatcher, new Pet(null, null, null, null, 650.0));

matchPetsFromTheList("Pets that are brown",colorMatcher, new Pet(null, null, null, "brown", 0.0));

method **matchPetsFromTheList** has a void return and three parameters – String criteria, PetMatcher matcher and Pet called myPet. Note: The second parameter allows you to send code as a parameter.

In this method you will first use matcher with firstPet to print out the first Pet.

Then create a List with all the matches using the matcher and print out the results.

Sample output for this example:

Pugs:

First: Cuddles: a black pug dog purchased for $1282.75

All matches:

Cuddles: a black pug dog purchased for $1282.75

Pets for $650 or less:

First: Max: a black poodle dog purchased for $540.50

All matches:

Max: a black poodle dog purchased for $540.50

HotDog: a brown dachsund dog purchased for $621.75

Slider: a brown garden snake purchased for $320.00

Pets that are brown:

First: Scruffy: a brown poodle dog purchased for $895.00

All matches:

Scruffy: a brown poodle dog purchased for $895.00

HotDog: a brown dachsund dog purchased for $621.75

Slider: a brown garden snake purchased for $320.00