Lambda Expressions

L.1

Functor.java | LengthFun.java

Source | History

```java
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package Lambda_1_a;

/**
 *
 * @author Carlos Guisao
 * @param <R>
 * @param <T>
 */
public interface Functor<R, T> {
    public R apply(T param);

}

```

Functor.java | LengthFun.java

Source | History

```java
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package Lambda_1_a;

/**
 *
 * @author Carlos Guisao
 */
public class LengthFun {

    public static void main(String[] args) {

        System.out.println("Example without using Lambda expressions");
        String test = "Carlos";
        System.out.println(test.length());

        System.out.println("Example using Lambda expresions");
        Functor<Integer, String> LengthFun = (x) -> x.length();
        System.out.println(LengthFun.apply("Carlos"));

    }
}
```

Lambda_1_a.LengthFun > main

Output

Debugger Console | Homework_5 (run)

```
run:
Example without using Lambda expressions
6
Example using Lambda expresions
6
BUILD SUCCESSFUL (total time: 0 seconds)
```

L.1 b

MyList.java | TimesTwoFun.java

Source | History

```java
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package Lambda_1_b;


/**
 *
 * @author Carlos Guisao
 */
public class TimesTwoFun {

    public int apply(int param)
    {
        return 2*param;
    }
}

```

MyList.java | TimesTwoFun.java

Source | History

```java
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package Lambda_1_b;

import Lambda_1_c.Functor2;
import Lambda_1_a.Functor;
import java.util.LinkedList;

/**
 *
 * @author Carlos Guisao
 * @param <T>
 */
public class MyList<T> extends LinkedList<T>{


    public <R> LinkedList map(Functor<R,T> fo){

        MyList<T> list = new MyList<>();

        MyList.this.forEach((item) -> {
            list.add((T) fo.apply(item));
        });

        return list;
    }

```

MyList.java | TimesTwoFun.java

Source | History

```java
44      public static void main(String[] args) {
45
46
47          System.out.println("TimesTwoFun function demostrated");
48          TimesTwoFun tt = new TimesTwoFun();
49          System.out.println(tt.apply(10));
50
51          System.out.println("Using Lambda expressions 10 * 2");
52
53          Functor<Integer, Integer> lambda = (x) -> x*2;
54
55          System.out.println(lambda.apply(10));
56
57          MyList<Integer> myInts = new MyList<>();
58          myInts.add(-2);
59          myInts.add(1);
60          myInts.add(0);
61          myInts.add(4);
62
63          System.out.println("Linkedlist before using the Functor functions");
64          System.out.println(myInts);
65
66          Functor<Integer, Integer> TimesTwoFun = (x) -> x*2;
67
68          System.out.println("Linkedlist after using the Functor functions");
69          System.out.println(myInts.map(TimesTwoFun));
70
71      }
72  }
```

Lambda_1_b.MyList > map > list

Output

Debugger Console | Homework_5 (run)

```
run:
TimesTwoFun function demostrated
20
Using Lambda expressions 10 * 2
20
Linkedlist before using the Functor functions
[-2, 1, 0, 4]
Linkedlist after using the Functor functions
[-4, 2, 0, 8]
BUILD SUCCESSFUL (total time: 0 seconds)
```

L.1 c

SummerTest.java | Summer.java | Functor2.java

Source | History

```java
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package Lambda_1_c;

/**
 *
 * @author carlo
 * @param <R>
 * @param <T1>
 * @param <T2>
 */
public interface Functor2<R,T1,T2>  {
        public R apply(T1 param1, T2 param2);
    }
```

SummerTest.java | Summer.java | Functor2.java

Source | History

```java
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package Lambda_1_c;

/**
 *
 * @author Carlos Guisao
 */
public class Summer implements Functor2{

    @Override
    public Object apply(Object param1, Object param2) {
        return (int)param1 + (int)param2;
    }
}
```

SummerTest.java | Summer.java | Functor2.java

Source | History

```java
public class SummerTest {
    public static void main(String[] args) {

        System.out.println("Summer function 5 + 6: ");
        Summer ref = new Summer();
        System.out.println(ref.apply(5, 6));

        LinkedList<Integer> myInts = new LinkedList<>();

        myInts.add(1);
        myInts.add(2);
        myInts.add(0);
        myInts.add(4);

        System.out.println(myInts);

        MyList list = new MyList();

        Functor2<Integer, Integer, Integer> variable = (x, y) -> x + y;

        System.out.println("Functor lambda expression example");

        System.out.println(variable.apply(5, 6));

        System.out.println("Functor lambda expression example adding "
                + "the variables in the list result");

        System.out.println(list.reduce(variable, 0, myInts));

    }
}
```

Lambda_1_c.SummerTest > main

Output

Debugger Console | Homework_5 (run)

```
run:
Summer function 5 + 6:
11
[1, 2, 0, 4]
Functor lambda expression example
11
Functor lambda expression example adding the variables in the list result
7
```

L.1 d

MRExample.java

```java
 *
 * @author Carlos Guisao
 */
public class MRExample {
    public static void main(String[] args) {

        MyList<String> ls = new MyList<>();

        ls.add("Patricia");
        ls.add("Andres");
        ls.add("Bruce");
        ls.add("Winny the poo");

        System.out.println(ls);

        Functor<Integer, String> LengthFun = (x) -> x.length();

        System.out.println("Total lenght of each string:");

        System.out.println(ls.map(LengthFun));

        MyList<Integer> lsInt = new MyList<>();

        System.out.println("Total number of chars in all strings");

        Functor2<Integer, Integer, Integer> sum = (x, y) -> x + y;

        System.out.println(lsInt.reduce(sum, 0, ls.map(LengthFun)));

    }
}
```

Output

Debugger Console | Homework_5 (run)

```
run:
[Patricia, Andres, Bruce, Winny the poo]
Total lenght of each string:
[8, 6, 5, 13]
Total number of chars in all strings
32
BUILD SUCCESSFUL (total time: 0 seconds)
```

L.1 e

MRExampleWithLambdas.java

Source History

```java
package Lambda_1_e;

import Lambda_1_a.Functor;
import Lambda_1_b.MyList;
import Lambda_1_c.Functor2;

/**
 *
 * @author Carlos Guisao
 */
public class MRExampleWithLambdas {
    public static void main(String[] args) {

        MyList<String> ls = new MyList<>();

        ls.add("Patricia");
        ls.add("Andres");
        ls.add("Bruce");
        ls.add("Winny the poo");

        System.out.println(ls);

        Functor<Integer, String> LengthFun = (x) -> x.length();
        Functor2<Integer, Integer, Integer> sum = (x, y) -> x + y;

        MyList<Integer> lsInt = new MyList<>();

        System.out.println(lsInt.reduce(sum, 0, ls.map(LengthFun)));

    }
}
```

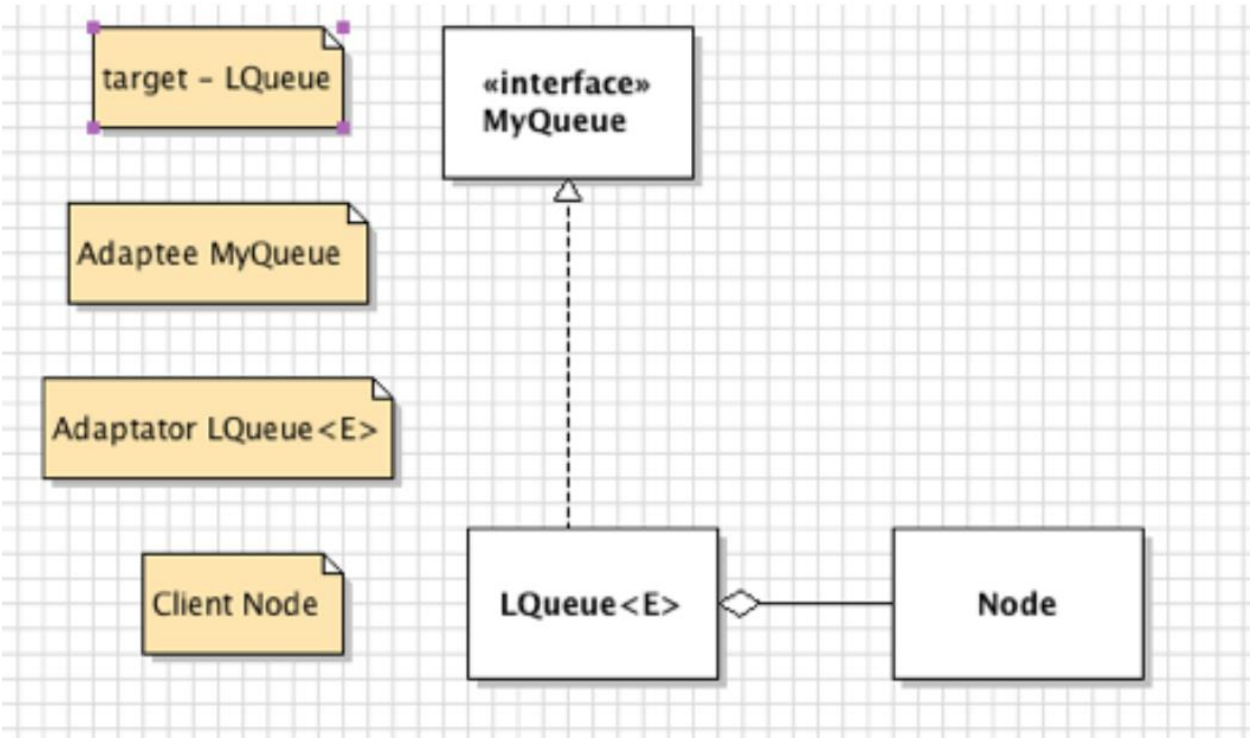
Output

Debugger Console | Homework_5 (run)

```
run:
[Patricia, Andres, Bruce, Winny the poo]
Total lenght of each string:
[8, 6, 5, 13]
Total number of chars in all strings
32
BUILD SUCCESSFUL (total time: 0 seconds)
```

10.1

a)


target - LQueue
«interface»
MyQueue
Adaptee MyQueue
Adaptator LQueue<E>
Client Node
LQueue<E>
Node

b)

LQueue.java | MyQueue.java | QueueTest.java

Source | History

```java
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package Chapter_10_1;

import java.util.Collection;
import java.util.LinkedList;
import java.util.NoSuchElementException;

/**
 *
 * @author Carlos Guisao
 * @param <E>
 */
public class LQueue <E> implements MyQueue <E> {

    public LQueue() {
        queue = new LinkedList<>();
    }

    @Override
    public void enqueue(E e) {
        queue.add(e);
    }

    @Override
    public void addAll(Collection<? extends E> c) {
        queue.addAll(c);
    }

    @Override
    public E head() {
        try {
            return queue.element();
        } catch (NoSuchElementException e) {
            throw new NoSuchElementException("queue is empty");
        }
    }
```

LQueue.java | MyQueue.java | QueueTest.java

Source | History

```java
            try {
                return queue.element();
            } catch (NoSuchElementException e) {
                throw new NoSuchElementException("queue is empty");
            }
        }

        @Override
        public E dequeue() {
            try {
                return queue.removeFirst();
            } catch (NoSuchElementException e) {
                throw new NoSuchElementException("queue is empty");
            }
        }

        @Override
        public int size() {
            return queue.size();
        }

        @Override
        public boolean isEmpty() {
            return queue.isEmpty();
        }

        private final LinkedList <E> queue;
    }

```

c)

LQueue.java | MyQueue.java | QueueTest.java

Source | History

```java
import java.util.LinkedList;

/**
 *
 * @author Carlos Guisao
 */
public class QueueTest {

    public static void main(String[] args) {
        LinkedList<Integer> integers = new LinkedList<>();
        for(int i=0; i<5; i++) {
            integers.add(i);
        }

        LQueue<Integer> queue = new LQueue<>();
        System.out.println("Adding list of integers to queue.");
        queue.addAll(integers); // 1
        System.out.println("Current size: " + queue.size() + "\n"); // 2

        System.out.println("Adding integer with enqueue.");
        queue.enqueue(5); // 3
        System.out.println("Current size: " + queue.size() + "\n");

        System.out.println("        Head: " + queue.head() + "\n"); // 4

        while (!queue.isEmpty()) { // 5
            System.out.println("     Dequeue: " + queue.dequeue()); // 6
            System.out.println("Current size: " + queue.size() + "\n");
        }
    }
}
```

LQueue.java | MyQueue.java | QueueTest.java

Source | History

```java
import java.util.LinkedList;

/**
 *
 * @author Carlos Guisao
 */
public class QueueTest {

    public static void main(String[] args) {
```

Chapter_10_1.QueueTest > main

Output

Debugger Console | Homework_5 (run)

```
run:
Adding list of integers to queue.
Current size: 5

Adding integer with enqueue.
Current size: 6

        Head: 0

     Dequeue: 0
Current size: 5

     Dequeue: 1
Current size: 4

     Dequeue: 2
Current size: 3

     Dequeue: 3
Current size: 2

     Dequeue: 4
Current size: 1

     Dequeue: 5
Current size: 0

BUILD SUCCESSFUL (total time: 0 seconds)
```

10.2

Stdout.java

```java

import java.io.PrintStream;

/**
 *
 * @author Carlos Guisao
 */
public class Stdout {

    public static void main(String[] args) {
        Stdout out = Stdout.getInstance();
        out.printLine("Hello, Carlos!!");
    }

    public void printLine(String s) {
        stream.println(s);
    }

    public static Stdout getInstance() {
        return instance;
    }

    private Stdout() {
        stream = System.out;
    }

    private final PrintStream stream;
    private static final Stdout instance = new Stdout();
}

```

Chapter_10_2.Stdout

Output

Debugger Console | Homework_5 (run)

```
run:
Hello, Carlos!!
BUILD SUCCESSFUL (total time: 0 seconds)
```

10.3

A - The differences between the decorator pattern and the proxy pattern is that the decorator pattern focuses on dynamically adding functions to an object where the proxy pattern focuses on controlling access to an object or when you want to “lazyinstantiate an object.

B- The MouseMotionAdapter class from the Swing library is not an adapter class in the sense of the Adapter design pattern because none of its method are implemented and left empty on purpose. This class is provided as a convenience for easily creating listeners by extending this class and overriding its methods.

7.1

Pair.java | PairTest.java | Utils.java

Source | History

```java
/**
 *
 * @author Carlos Guisao
 */
public class PairTest {
    public static void main(String[] args) throws Exception {
        ArrayList<Pair> pairs = deserializePairs();
        showPairData(pairs);
        serializePairs(pairs);
    }

    /**
     * Reads a file of serialized Pairs and generates an ArrayList of them.
     * If no serialized Pairs exist yet, random Pairs are generated.
     * @return An ArrayList of Pairs
     * @throws Exception on input stream errors
     * @see #serializePairs(ArrayList)
     */
    private static ArrayList<Pair> deserializePairs() throws Exception {
        FileInputStream fis;
        ObjectInputStream ois;
        ArrayList<Pair> pairs;
        try {
            fis = new FileInputStream("object.ser");
            ois = new ObjectInputStream(fis);
            pairs = new ArrayList<>();
```

Pair.java | PairTest.java | Utils.java

Source | History

```java
     * @throws Exception on input stream errors
     * @see #serializePairs(ArrayList)
     */
    private static ArrayList<Pair> deserializePairs() throws Exception {
        FileInputStream fis;
        ObjectInputStream ois;
        ArrayList<Pair> pairs;
        try {
            fis = new FileInputStream("object.ser");
            ois = new ObjectInputStream(fis);
            pairs = new ArrayList<>();
            boolean done = false;
            while (!done) {
                try {
                    pairs.add((Pair) ois.readObject());
                } catch (EOFException e) {
                    done = true;
                }
            }
            ois.close();
            fis.close();
        } catch (FileNotFoundException e) {
            pairs = Utils.randomPairsWithClone();
        }
        return pairs;
    }
```

Pair.java | PairTest.java | Utils.java

Source | History

```java
     *       ordered such that the first and third Pair in the ArrayList
     *       are clones.
     * @see Utils randomPairsWithClone()
     */
    private static void showPairData(ArrayList<Pair> pairs) {
        pairs.stream().map((Pair pair) -> {
            System.out.println("      Key: " + pair.k());
            return pair;
        }).map((pair) -> {
            System.out.println("    Value: " + pair.v());
            return pair;
        }).map((pair) -> {
            System.out.println("To String: " + pair.toString());
            return pair;
        }).forEachOrdered((pair) -> {
            System.out.println("Hash Code: " + pair.hashCode() + "\n");
        });

        System.out.println("First ==  Third: " + pairs.get(0).equals(pairs.get(2))); // should be true
        System.out.println("First == Second: " + pairs.get(0).equals(pairs.get(1)));// should be false
    }
```

Pair.java | PairTest.java | Utils.java

Source | History

```java
        System.out.println("First == Third: " + pairs.get(0).equals(pairs.get(2))); // should be true
        System.out.println("First == Second: " + pairs.get(0).equals(pairs.get(1)));// should be false
    }

    /**
     * Serializes Pairs and saves them to a file.
     * @param pairs Pairs to be serialized.
     * @throws Exception on output stream errors
     * @see #deserializePairs()
     */
    private static void serializePairs(ArrayList<Pair> pairs) throws Exception {
        try (FileOutputStream fos = new FileOutputStream("object.ser");
                ObjectOutputStream oos = new ObjectOutputStream(fos)) {
            for (Pair pair : pairs) {
                oos.writeObject(pair);
            }
        }
    }
}
```

Pair.java | PairTest.java | Utils.java

Source | History

```java
import java.util.Collection;
import java.util.Collections;
import java.util.Random;

/**
 *
 * @author Carlos Guisao
 */
public class Utils {

    public static void main(String[] args) {
        ArrayList<Pair> pairs = randomPairs(),
                        sorted = sortPairCollection(pairs);

        pairs.forEach((pair) -> {
            System.out.println(pairString(pair));
        });

        System.out.println();

        sorted.forEach((pair) -> {
            System.out.println(pairString(pair));
        });
    }
```

Pair.java | PairTest.java | Utils.java

Source | History

```java
     * Sorts a collection of Pairs and returns them in an ArrayList
     * @param original Collection containing the Pairs to be sorted
     * @return An ArrayList of sorted Pairs
     * Note: This method is generic even though generic types do not
     *       need to be referred to explicitly in the method. This method
     *       works on any Pair regardless of the type of key or value.
     */
    public static ArrayList<Pair> sortPairCollection(Collection<Pair> original) {
        ArrayList<Pair> pairList = new ArrayList<>(original);
        Collections.sort(pairList);
        return pairList;
    }

    public static ArrayList<Pair> randomPairsWithClone() throws CloneNotSupportedException {
        ArrayList<Pair> pairs = new ArrayList<>();
        pairs.add(randomIntPair());
        pairs.add(randomIntPair());
        pairs.add((Pair) pairs.get(0).clone()); // always have one clone so we can test it
        return pairs;
    }

    public static Pair randomIntPair() {
        Random random = new Random();
        return new Pair<>(random.nextInt(100), random.nextInt(100));
    }

```

Pair.java | PairTest.java | Utils.java

Source | History

```java
    public static ArrayList<Pair> randomPairs() {
        ArrayList<Pair> pairs = new ArrayList<>();
        pairs.add(randomIntPair());
        pairs.add(randomIntPair());
        pairs.add(randomIntPair());
        pairs.add(randomIntPair());
        pairs.add(randomIntPair());
        pairs.add(randomIntPair());
        return pairs;
    }

    public static String pairString(Pair pair) {
        return "("+pair.k()+", "+pair.v()+")";
    }
    private Utils() {}
}

```

Pair.java | PairTest.java | Utils.java

Source | History

```java
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package Chapter_7;

import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.io.Serializable;

/**
 *
 * @author Carlos Guisao
 */
public class Pair<K extends Comparable<K>, V extends Comparable<V>>
        implements Cloneable, Serializable, Comparable<Pair<K,V>> {

    public Pair(K k, V v) {
        key = k;
        val = v;
    }

    public K k() {return key;}
    public V v() {return val;}
```

Pair.java | PairTest.java | Utils.java

Source | History

```java
    public boolean equals(Object o) {
        if (o == null) return false;
        if (!getClass().equals(o.getClass())) return false;
        final Pair<K,V> other = (Pair<K,V>) o;
        return key.equals(other.k()) && val.equals(other.v());
    }

    public int hashCode() {
        return key.hashCode() + val.hashCode();
    }

    public String toString() {
        return this.getClass().getName()+"["+key.toString()+", "+val.toString()+"]";
    }

    public Pair<K,V> clone() {
        return new Pair<>(k(), v());
    }

    public int compareTo(Pair<K, V> other) {
        if (key.compareTo(other.k()) == 0)
            return val.compareTo(other.v());
        return key.compareTo(other.k());
    }

```

Pair.java | PairTest.java | Utils.java

Source | History

```java
    }

    private void writeObject(ObjectOutputStream os) throws Exception {
        os.defaultWriteObject();
        os.writeObject(key);
        os.writeObject(val);
    }

    private void readObject(ObjectInputStream os) throws Exception {
        os.defaultReadObject();
        key = (K) os.readObject();
        val = (V) os.readObject();
    }

    private K key;
    private V val;
}

```

Output

Debugger Console | Homework_5 (run)

```
run:
      Key: 96
    Value: 60
To String: Chapter_7.Pair[96, 60]
Hash Code: 156

      Key: 14
    Value: 36
To String: Chapter_7.Pair[14, 36]
Hash Code: 50

      Key: 96
    Value: 60
To String: Chapter_7.Pair[96, 60]
Hash Code: 156

First ==  Third: true
First == Second: false
BUILD SUCCESSFUL (total time: 0 seconds)
```