JAVA TRAINING CENTER

Java Training Center

(No.1 in Training & placement)

Java 8(LAMBDA)

WorkBook

Master the Content…

Subscribe :

www.youtube.com/javatrainingcenterjtc

Author

Som Prakash Rai

JAVA TRAINING CENTER

Jtc 1 – Lambda Expression (Functional Interface)

```
package com.jtc.p1;
interface Executable {
Integer execute(Integer a, Integer b);
}
interface StringExecutable {
Integer execute(String a);
}
class Runner {
public void run(Executable e) {
System.out.println("Entered into run method");
Integer value = e.execute(12, 11);
System.out.println("Return Value is" + value);
System.out.println("Test1");
}
public void run(StringExecutable e) {
System.out.println("Entered into run method");
Integer value = e.execute("Hello");
System.out.println("Return Value is" + value);
}
}

public class Jtc1 {
/*
* @Author    : Som Prakash Rai
* @Join      : Java Training Center
* @visit     : www.jtcindia.org
*@Subscribe : youtube.com/javatrainingcenterjtc
* */
public static void main(String[] args) {
int d = 10;
Runner runner = new Runner();

/*
* runner.run(new Executable() { public Integer execute(Integer a,
* Integer b) { // can do this // methods of // annonumous // classes
* int d=8; System.out.println("Value of D is" + d); ;
* System.out.println("Hello There"); return a + b; } });
```

```
*/
System.out.println("*******************************\n");
// runner.run(() -> System.out.println("Hello there"));
runner.run((a, b) -> {
// cannot do this; no new scope. int d=10;
return 8 + a + b;
});
System.out.println("*******************************\n");
Executable ex = (a, b) -> {
// cannot do this; no new scope. int d=10;
System.out.println("ooh");
return 8 + a + b;
};
runner.run(ex);
System.out.println("*******************************\n");
Object codeBlock = (Executable) (a, b) -> {
// cannot do this; no new scope. int d=10;
System.out.println("ooh");
return 8 + a + b;
};
System.out.println("codeBlock" + codeBlock);
// runner.run(codeBlock);
}}
```

Jtc 2 – Lambda Expression (Functional Interface)

```
package com.jtc.p2;
public class HelloLambda {


public interface HelloType {
void hello(String text, String name);
// void hai(int a);
}
public interface HaiType {
void main(int a);
}
public static void main(String[] args) {
HelloType helloLambda = (String text1, String name) -> {
System.out.println("Hello " + text1);
};
```

```
HaiType haitypes = (int a1) -> {
System.out.println("HaiTypes   :" + a1);
//return 10;
};
HaiType ht = (int b1) -> {
System.out.println("  ");
};
// Invoke the method call
helloLambda.hello("Lambda", "SOM");
System.out.println(helloLambda);
haitypes.main(111);
}
}
```

Jtc 3 – Lambda Expression (Functional Interface- Method with Different Return type and different parameters)

```
public class Jtc3 {
// Functional interface returning a string

interface StringReturn {
String returnMessage();
}
// Functional interface returning an int
interface ActionCode {
int returnCode(String codestr);
}
 // example of a lambda containing no arguments
public static void noArguments() {
StringReturn msg = () -> "This is a test";
System.out.println(msg.returnMessage());
}
public static void returnCode() {
ActionCode code = (codestr) -> {
switch (codestr) {
case "ACTIVE":
return 0;
case "INACTIVE":
return 1;
default:
return -1;
}
```

```
};
System.out.println("Returns: " + code.returnCode("ACTIVE"));
}
public static void main(String[] args) {
noArguments();
returnCode();
}}
```

Jtc 4 – Lambda Expression (Reverse of String)

```
package com.jtc.p3;
import java.util.function.Function;
public class Reverse {
@FunctionalInterface
interface ReverseType {
String reverse(String text);
}
public static void main(String[] args) {
ReverseType newText = (testText) -> {
String tempStr = "";
for (String part : testText.split(" ")) {
tempStr = new StringBuilder(part).reverse().toString();
}
return tempStr;
};

Function<String, String> newText2 = (testText) -> {
String tempStr = "";
for (String part : testText.split(" ")) {
tempStr = new StringBuilder(part).reverse().toString();
}
return tempStr;
};
System.out.println(newText.reverse("HELLO"));
System.out.println(newText2.apply("JTC"));
}
}
```

Jtc 4 – Lambda Expression (Shorting in different Ways)

```
package com.jtc.p4;
public class Player {
```

```
private String firstName = null;
private String lastName = null;
private String position = null;
private int status = -1;
private int goals;

public Player() {

}

public Player(String position, int status) {
this.position = position;
this.status = status;
}

protected String playerStatus() {
String returnValue = null;

switch (getStatus()) {
case 0:
returnValue = "ACTIVE";
case 1:
returnValue = "INACTIVE";
case 2:
returnValue = "INJURY";
default:
returnValue = "ON_BENCH";
}
return returnValue;
}
public String playerString() {
return getFirstName() + " " + getLastName() + " - " + getPosition();
}
//Getters and Setters
}
package com.jtc.p4;
import java.util.ArrayList;
import java.util.Collections;
import java.util.Comparator;
import java.util.List;
public class Sorter {
static List<Player> team;
```

```
private static void loadTeam() {
System.out.println("Loading team...");

team = new ArrayList();
System.out.println("1.********************");
Player player1 = new Player();
player1.setFirstName("Som");
player1.setLastName("Rai");
player1.setGoals(5);

System.out.println("2.********************");

Player player2 = new Player();
player2.setFirstName("Prakash");
player2.setLastName("Rai");
player2.setGoals(15);

System.out.println("3.********************");

Player player3 = new Player();
player3.setFirstName("Manish");
player3.setLastName("ahuja");
player3.setGoals(1);
System.out.println("4.********************");

Player player4 = new Player();
player4.setFirstName("Rai");
player4.setLastName("Prakash");
player4.setGoals(18);
System.out.println("5.********************");

Player player5 = new Player();
player5.setFirstName("Ramesh");
player5.setLastName("pandey");
player5.setGoals(7);

team.add(player1);
team.add(player2);
team.add(player3);
team.add(player4);
team.add(player5);
```

```
}

public static void main(String[] args) {
// Load team list
loadTeam();
Comparator<Player> byGoals = Comparator.comparing(Player::getGoals);
System.out.println("== Sort by Number of Goals ==");
team.stream().sorted(byGoals).map(p -> p.getFirstName() + " " + p.getLastName()
+ " - " + p.getGoals())
.forEach(element -> System.out.println(element));
System.out.println("== Sort by Last Name ==\n");
Collections.sort(team, (p1, p2) -> p1.getLastName().compareTo(p2.getLastName()));
team.stream().forEach((p) -> {
System.out.println(p.getLastName());
});
}}
```

Jtc 5 – Lambda Expression (Runnable Interface Implementation)

```
package com.jtc.p6;
public class LambdaRunnable {
public static void main(String[] args) {
Runnable oldRunnable = new Runnable() {
@Override
public void run() {
int x = 5 * 3;
System.out.println("The variable using the old way equals: " + x);
}
};
Runnable lambdaRunnable = () -> {
int x = 5 * 3;
System.out.println("The variable using the lambda equals: " + x);
};
oldRunnable.run();
lambdaRunnable.run();}}
```