By – Pankaj Ku Chouhan

Batch -2

**Lambda Expression**

The Lambda expression is used to provide the implementation of an interface which has functional interface. It saves a lot of code.

**Functional Interface**

1. @FunctionalInterface  //It is optional
2. **interface** Drawable{
3. **public** **void** draw();
4. }
6. **public** **class** LambdaExpressionExample2 {
7. **public** **static** **void** main(String[] args) {
8. **int** width=10;
10. //with lambda
11. Drawable d2=()->{
12. System.out.println("Drawing "+width);
13. };
14. d2.draw();
15. }
16. }

Java Lambda Expression Example: No Parameter

1. **interface** Sayable{
2. **public** String say();
3. }
4. **public** **class** LambdaExpressionExample3{
5. **public** **static** **void** main(String[] args) {
6. Sayable s=()->{
7. **return** "I have nothing to say.";
8. };
9. System.out.println(s.say());
10. }
11. }

Java Lambda Expression Example: Single Parameter

1. **interface** Sayable{
2. **public** String say(String name);
3. }
5. **public** **class** LambdaExpressionExample4{
6. **public** **static** **void** main(String[] args) {
8. // Lambda expression with single parameter.
9. Sayable s1=(name)->{
10. **return** "Hello, "+name;
11. };
12. System.out.println(s1.say("Sonoo"));
14. // You can omit function parentheses
15. Sayable s2= name ->{
16. **return** "Hello, "+name;
17. };
18. System.out.println(s2.say("Sonoo"));
19. }
20. }

## Java Lambda Expression Example: Multiple Parameters

1. **interface** Addable{
2. **int** add(**int** a,**int** b);
3. }
5. **public** **class** LambdaExpressionExample5{
6. **public** **static** **void** main(String[] args) {
8. // Multiple parameters in lambda expression
9. Addable ad1=(a,b)->(a+b);
10. System.out.println(ad1.add(10,20));
12. // Multiple parameters with data type in lambda expression
13. Addable ad2=(**int** a,**int** b)->(a+b);
14. System.out.println(ad2.add(100,200));
15. }
16. }

Java Lambda Expression Example: with or without return keyword

In Java lambda expression, if there is only one statement, you may or may not use return keyword. You must use return keyword when lambda expression contains multiple statements.

1. **interface** Addable{
2. **int** add(**int** a,**int** b);
3. }
5. **public** **class** LambdaExpressionExample6 {
6. **public** **static** **void** main(String[] args) {
8. // Lambda expression without return keyword.
9. Addable ad1=(a,b)->(a+b);
10. System.out.println(ad1.add(10,20));
12. // Lambda expression with return keyword.
13. Addable ad2=(**int** a,**int** b)->{
14. **return** (a+b);
15. };
16. System.out.println(ad2.add(100,200));
17. }
18. }