**MULTIPLE CHOICE QUESTION.**

**JAVA LAMBDA TRAINING 26th Sep to 30th Sep 2016.**

**RAJESH PARAB**

1. What is the syntax for a lambda expression?
2. <return type> <method name with argument> <throw clause> <method body>
3. <method name with argument> “->” symbol <method body>
4. <parameter list with or without parenthesis> “->” symbol and <code block with or without curly braces>
5. <parameter with data type> <code block with curly braces>

**ANSWER – C**

1. Which of the followings are valid lambda expressions?
2. () -> 42;
3. x -> x + 1;
4. (int x, int y) -> x + y;
5. All of the above

**ANSWER – D**

1. Which of the followings are invalid lambda expressions?
2. () -> 42;
3. x -> x + 1;
4. int x, int y -> x + y;
5. None of the above

**ANSWER – C**

1. Which of the following is/are true?
2. Lambdas expressions can be recursive
3. Lambda expressions can be generics.
4. Both 1 and 2 are true.
5. Both 1 and 2 are false.
6. Only 1 is true
7. Only 2 is true

**ANSWER – B**

1. Which of the following is not functional interface
2. Interface Compose{

void apply();

}

1. Interface Compose{

void apply();

default calculate { .. }

1. @FunctionalInterface

Interface Compose{

void apply();

void calculate (int discount);

}

1. Interface Compose{

void apply();

}

@FunctionalInterface

Interface MyComposer extends Compose {

default calculate();

}

**ANSWER – C**

1. Which of the followings is/are true about functional interface?
2. An interface with one abstract method
3. Yes, it can have default methods and methods inherited from Object
4. @FunctionalInterface annotation is optional
5. All of the above

**ANSWER – D**

1. Which of the followings is/are true about @Functional Interface annotation?
2. It indicates that an interface is intended as a functional interface and triggers certain compiler checks.
3. It is mandatory.
4. A and B both are true.
5. Only A is true.

**ANSWER – D**

1. Which of the followings lambda expression does not match the target type of Predicate functional interface?

A) Predicate<String> f = str -> str !=null;

1. Predicate<String> f = (String str) -> { return str !=null; };

C) Predicate<String>f = (String str) -> “welcome”+ str;

D) Predicate<String> = str -> true;.

**ANSWER – C**

1. Which of the followings lambda expression does not match the target type of Consumer functional interface?

A) Consumer <String> f = str -> System.out.println(str);

1. Consumer <String> f = (String str) -> { return str ; };

C) Consumer <String>f = (String str) -> “welcome”+ str;

D) Consumer <String> = str -> { int i=str.length(); };

**ANSWER – B C**

1. Which of the followings lambda expression does not match the target type of Supplier functional interface?

A) Supplier <String> f = () -> "hello";

1. Supplier <Employee> f = () -> new Employee();

C) Supplier <Integer>f = () -> 10+20;

D) Supplier <String> = () -> str.length();

**ANSWER – D**

1. Which of the followings lambda expression does not match the target type of BinaryOperator functional interface?

A) BinaryOperator <String> f = (str1,str2) -> str1+str2;

1. BinaryOperator <String> f = (str1,str2) -> new

StringBuilder(str1).toString()

C) BinaryOperator <String> f = (str1,str2) ->

System.out.println(str1);

D) BinaryOperator <String> f = (str1,str2) -> str1;

**ANSWER – C**

1. Which of the followings reference to constructor is/are correct?

A) String::new

B) ArrayList<String>::new

C) String[]::new

D) All of the above

**ANSWER – D**

1. Which of the followings reference to method is/are incorrect?

A) String::length

B) Collection::toArray

C) System::out.println

D) System.out::println

**ANSWER – C**

1. What are default methods used for?

A) Evolution of existing interfaces plus API development in general

B) To serve as target type for lambda expressions.

C) Served as default implementation which cannot be override;

D) None of the above

**ANSWER – A**

1. The meanings of the keywords this and super are the same inside the lambda expression and its enclosing method

A) True

B) False

**ANSWER – A**