Lambda Expression

Agenda:

Lambda Expression
Regular Expression

Md. Mamun Hossain
B.Sc. (Engg.) & M.Sc. (Thesis) in CSE , SUST
Assistant Professor, Dept. of CSE, BAUST

Lambda Expression is used to provide the implementation of a functional interface

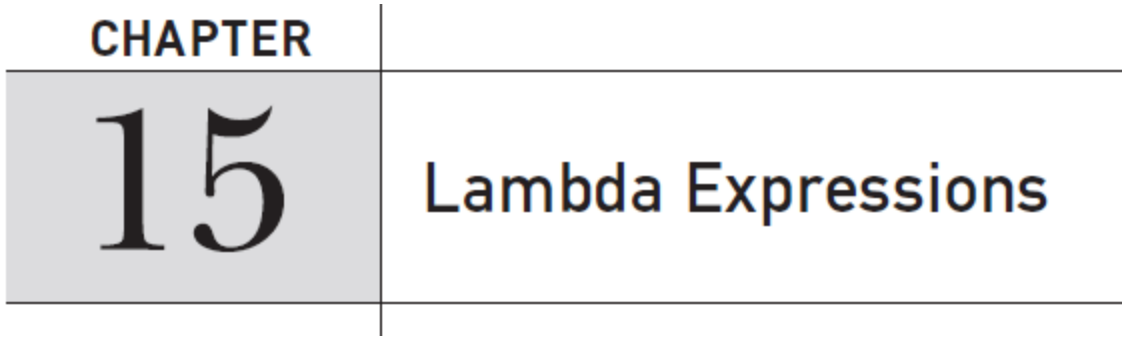
Lambda Expression

*used to provide the implementation of a **functional interface***
*(**functional interface** : an interface which has an abstract method only)*

Md. Mamun Hossain
B. Sc. (Engg.) & M. Sc. (Thesis) in CSE, SUST
Asst. Professor, Dept. of CSE , BAUST

Reference Text : Chap. -15

Lambda Expression



Keywords:

Lambda Expression basis

Functional Interface

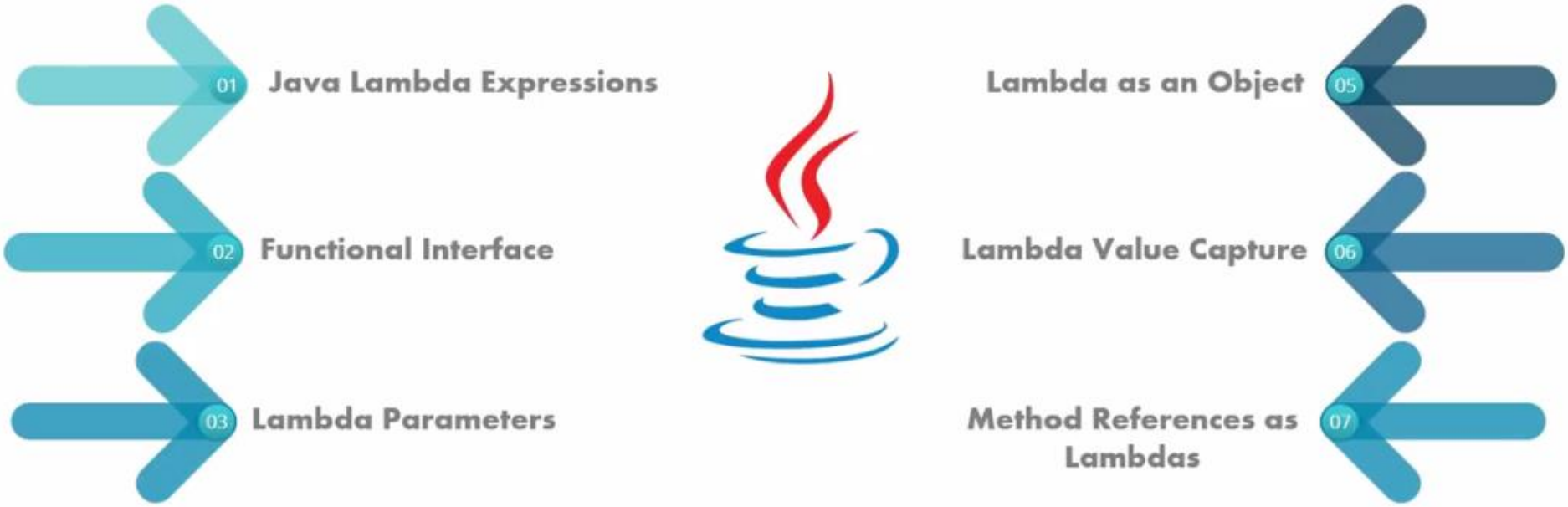
Lambda Expression Syntax

Anonymous class

Lambda Expression and Parameter

Md. Mamun Hossain
B. Sc. (Engg.) & M. Sc. (Thesis) in CSE, SUST
Asst. Professor, Dept. of CSE , BAUST

Topics to Discuss



Md. Mamun Hossain
B. Sc. (Engg.) & M. Sc. (Thesis) in CSE, SUST
Asst. Professor, Dept. of CSE , BAUST

Lambda Expressions: Basics

- Lambda expression is a new and important feature of Java
- Lambda Expressions were added in Java 8.
- *A lambda expression is a short block of code which takes in parameters and returns a value.*
- *Lambda expressions are* ***similar to methods***, but they ***do not need a name*** and they can be implemented right in the body of a method.

Md. Mamun Hossain
B. Sc. (Engg.) & M. Sc. (Thesis) in CSE, SUST
Asst. Professor, Dept. of CSE , BAUST

Lambda Expressions: Syntax

Syntax

The simplest lambda expression contains a single parameter and an expression:

```
parameter -> expression
```

To use more than one parameter, wrap them in parentheses:

```
(parameter1, parameter2) -> expression
```

Md. Mamun Hossain
B. Sc. (Engg.) & M. Sc. (Thesis) in CSE, SUST
Asst. Professor, Dept. of CSE , BAUST

Lambda Expressions: Expression

```
parameter -> expression
```

```
(parameter1, parameter2) -> expression
```

Expressions are limited. They have to immediately return a value, and they cannot contain variables, assignments or statements such as if or for. In order to do more complex operations, a code block can be used with curly braces. If the lambda expression needs to return a value, then the code block should have a return statement.

```
(parameter1, parameter2) -> { code block }
```

Md. Mamun Hossain
B. Sc. (Engg.) & M. Sc. (Thesis) in CSE, SUST
Asst. Professor, Dept. of CSE , BAUST

Lambda Expressions : Advantages

- Lambda expression provides a clear and concise way to represent one method interface using an expression.
- It is very useful in collection library. It helps to iterate, filter and extract data from collection.
- *The Lambda expression is used to provide the implementation of an interface which has* ***functional interface****.*
- It saves a lot of code. In case of lambda expression, we don't need to define the method again for providing the implementation. Here, we just write the implementation code.
- Java lambda expression is treated as a function, so compiler does not create .class file.

Md. Mamun Hossain
B. Sc. (Engg.) & M. Sc. (Thesis) in CSE, SUST
Asst. Professor, Dept. of CSE , BAUST

Functional Interface

- Lambda expression provides implementation of *functional interface*. An interface which has only one abstract method is called functional interface.
- Java provides an anotation *@FunctionalInterface*, which is used to declare an interface as functional interface.

```
@FunctionalInterface
interface Cab{ // When an interface will have excatly 1 abstract method
    void bookCab(); // -> by default public abstract void bookCab();
}
```

Why use Lambda Expression

- ✓ To provide the implementation of Functional interface.
- ✓ Less coding.

Md. Mamun Hossain
B. Sc. (Engg.) & M. Sc. (Thesis) in CSE, SUST
Asst. Professor, Dept. of CSE , BAUST

Lambda Expression : Syntax Explain

```
(argument-list) -> {body}
```

Java lambda expression is consisted of three components.

1) Argument-list: It can be empty or non-empty as well.

2) Arrow-token: It is used to link arguments-list and body of expression.

3) Body: It contains expressions and statements for lambda expression.

Md. Mamun Hossain
B. Sc. (Engg.) & M. Sc. (Thesis) in CSE, SUST
Asst. Professor, Dept. of CSE , BAUST

Java Lambda Expression: Example

```
interface Drawable{
    public void draw();
}
public class LambdaExpressionExample {
    public static void main(String[] args) {
        int width=10;

        //without lambda, Drawable implementation using anonymous class
        Drawable d=new Drawable(){
            public void draw(){System.out.println("Drawing "+width);}
        };
        d.draw();
    }
}
```

```
@FunctionalInterface  //It is optional
interface Drawable{
    public void draw();
}

public class LambdaExpressionExample2 {
    public static void main(String[] args) {
        int width=10;

        //with lambda
        Drawable d2=()->{
            System.out.println("Drawing "+width);
        };
        d2.draw();
    }
}
```

Md. Mamun Hossain
B. Sc. (Engg.) & M. Sc. (Thesis) in CSE, SUST
Asst. Professor, Dept. of CSE , BAUST

```java
@FunctionalInterface
interface Cab{ // When an interface will have excatly 1 abstract method we can say it as a Functional I
    void bookCab(); // -> by default public abstract void bookCab();
}

/*class UberX implements Cab{
    public void bookCab(){
        System.out.println("UberX Booked !! Arriving Soon !!");
    }     public class LambdaApp {
}*/
        public static void main(String[] args) {

            // 1.
            //Cab cab = new UberX(); // Polymorphic Statement
            //cab.bookCab();

            // 2.
            // Anonymous Class Implementation
            /*Cab cab = new Cab() {
                @Override
                public void bookCab() {
                    System.out.println("UberX Booked !! Arriving Soon !!");
                }
            };

            cab.bookCab();*/

            // 3.
            // Using a Lambda Expression
            Cab cab =()->{
                System.out.println("UberX Booked !! Arriving Soon !!");
            };

            cab.bookCab();
```

Mamun Hossain
is) in CSE, SUST
of CSE , BAUST

Lambda Expression & Parameter

```
interface Sayable{
    public String say();
}
public class LambdaExpressionExample3{
public static void main(String[] args) {
    Sayable s=()->{
        return "I have nothing to say.";
    };
    System.out.println(s.say());
}
}
```

```
interface Sayable{
    public String say(String name);
}

public class LambdaExpressionExample4{
    public static void main(String[] args) {

        // Lambda expression with single parameter.
        Sayable s1=(name)->{
            return "Hello, "+name;
        };
        System.out.println(s1.say("Sonoo"));

        // You can omit function parentheses
        Sayable s2= name ->{
            return "Hello, "+name;
        };
        System.out.println(s2.say("Sonoo"));
    }
}
```

ossain
, SUST
Asst. Professor, Dept. of CSE , BAUST

Java Lambda Parameters

Lambda Expressions can take parameters just like methods

Zero Parameters

```
() -> System.out.println("Zero parameter lambda");
```

One Parameter

```
(param) -> System.out.println("One parameter: " + param);
```

Multiple Parameters

```
(p1, p2) -> System.out.println("Multiple parameters: " +
  p1 + ", " + p2);
```

Md. Mamun Hossain
B. Sc. (Engg.) & M. Sc. (Thesis) in CSE, SUST
Asst. Professor, Dept. of CSE , BAUST

Lambda Expression Example Multiple Parameters

```
interface Addable{
    int add(int a,int b);
}

public class LambdaExpressionExample5{
    public static void main(String[] args) {

        // Multiple parameters in lambda expression
        Addable ad1=(a,b)->(a+b);
        System.out.println(ad1.add(10,20));

        // Multiple parameters with data type in lambda expression
        Addable ad2=(int a,int b)->(a+b);
        System.out.println(ad2.add(100,200));
    }
}
```

Md. Mamun Hossain
B. Sc. (Engg.) & M. Sc. (Thesis) in CSE, SUST
Asst. Professor, Dept. of CSE , BAUST

Lambda Expression Example: with or without return keyword

```
interface Addable{
   int add(int a,int b);
}

public class LambdaExpressionExample6 {
   public static void main(String[] args) {

      // Lambda expression without return keyword
      Addable ad1=(a,b)->(a+b);
      System.out.println(ad1.add(10,20));

      // Lambda expression with return keyword.
      Addable ad2=(int a,int b)->{
                     return (a+b);
                     };
      System.out.println(ad2.add(100,200));
   }
}
```

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

Md. Mamun Hossain
B. Sc. (Engg.) & M. Sc. (Thesis) in CSE, SUST
Asst. Professor, Dept. of CSE , BAUST

Lambda Expression Example: Multiple Statèrent

```
@FunctionalInterface
interface Sayable{
    String say(String message);
}

public class LambdaExpressionExample8{
    public static void main(String[] args) {

        // You can pass multiple statements in lambda expression
        Sayable person = (message)-> {
            String str1 = "I would like to say, ";
            String str2 = str1 + message;
            return str2;
        };
            System.out.println(person.say("time is precious."));
    }
}
```

In You must use return keyword when lambda expression contains multiple statements.

Md. Mamun Hossain
B. Sc. (Engg.) & M. Sc. (Thesis) in CSE, SUST
Asst. Professor, Dept. of CSE , BAUST

Lambda Expression Example: Creating Thread

```
public class LambdaExpressionExample9{
    public static void main(String[] args) {

        //Thread Example without lambda
        Runnable r1=new Runnable(){
            public void run(){
                System.out.println("Thread1 is running...");
            }
        };
        Thread t1=new Thread(r1);
        t1.start();
        //Thread Example with lambda
        Runnable r2=()->{
                System.out.println("Thread2 is running...");
        };
        Thread t2=new Thread(r2);
        t2.start();
    }
}
```

You can use lambda expression to run thread. In the following example, we are implementing run method by using lambda expression.

Md. Mamun Hossain
B. Sc. (Engg.) & M. Sc. (Thesis) in CSE, SUST
Asst. Professor, Dept. of CSE , BAUST

Java Lambda Expression Example: Foreach Loop

```
import java.util.*;
public class LambdaExpressionExample7{
    public static void main(String[] args) {

        List<String> list=new ArrayList<String>();
        list.add("ankit");
        list.add("mayank");
        list.add("irfan");
        list.add("jai");

        list.forEach(
            (n)->System.out.println(n)
        );
    }
}
```

Md. Mamun Hossain
Sc. (Engg.) & M. Sc. (Thesis) in CSE, SUST
Asst. Professor, Dept. of CSE , BAUST

Lambda as an Object

A Java lambda expression is essentially an object that can be assigned to a variable and passed around

INTERFACE

```
public interface LambdaComparator {
public boolean compare(int a1, int a2);
}
```

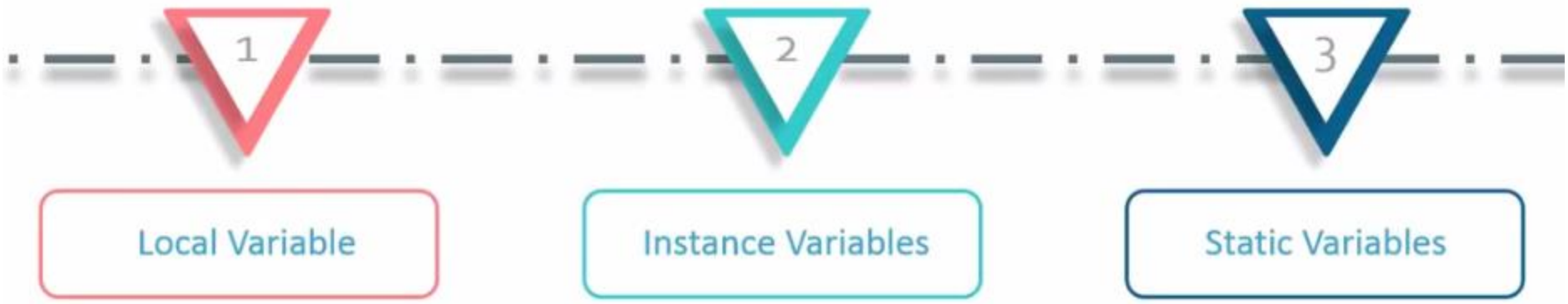
IMPLEMENTING CLASS

```
LambdaComparator myComparator = (a1, a2) -> return a1 > a2;

boolean result = myComparator.compare(2, 5);
```

Md. Mamun Hossain
B. Sc. (Engg.) & M. Sc. (Thesis) in CSE, SUST
Asst. Professor, Dept. of CSE , BAUST

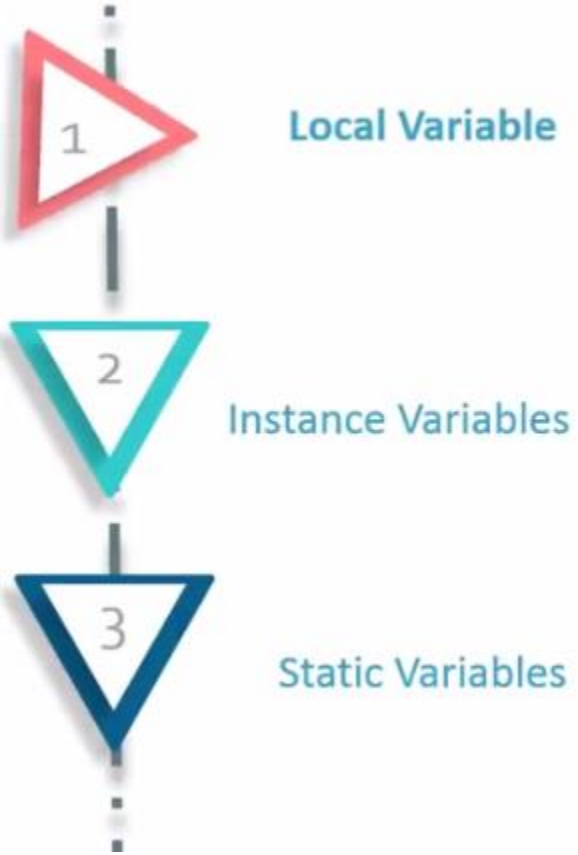
Lambda Expression: Variable Capture

Java lambda expression can access variables that are declared outside the lambda function body under certain circumstances



Md. Mamun Hossain
B. Sc. (Engg.) & M. Sc. (Thesis) in CSE, SUST
Asst. Professor, Dept. of CSE , BAUST

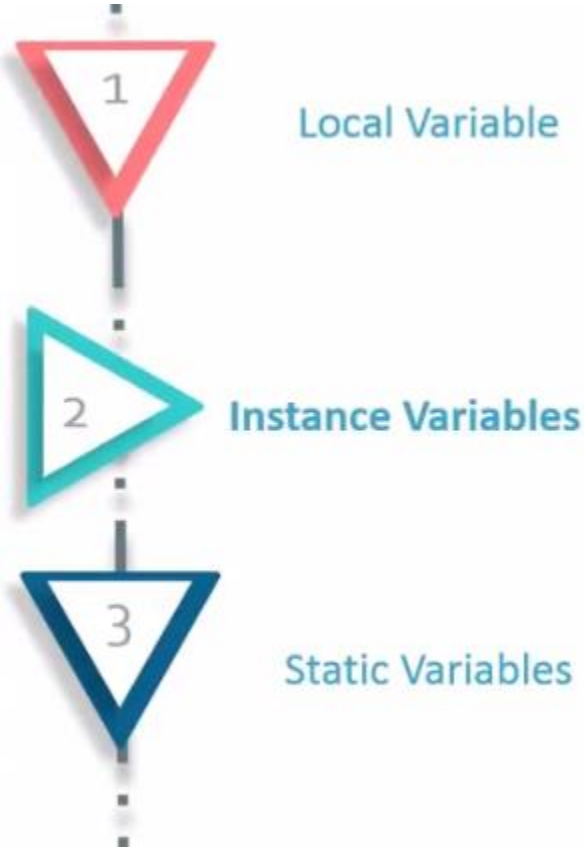
Lambda Expression: Variable Capture




```
String myStr = "Welcome to Edureka!";

MyLambda dis = (chars) -> {
    return myStr + ":" + new String(chars);
};
```

Md. Mamun Hossain
B. Sc. (Engg.) & M. Sc. (Thesis) in CSE, SUST
Asst. Professor, Dept. of CSE , BAUST

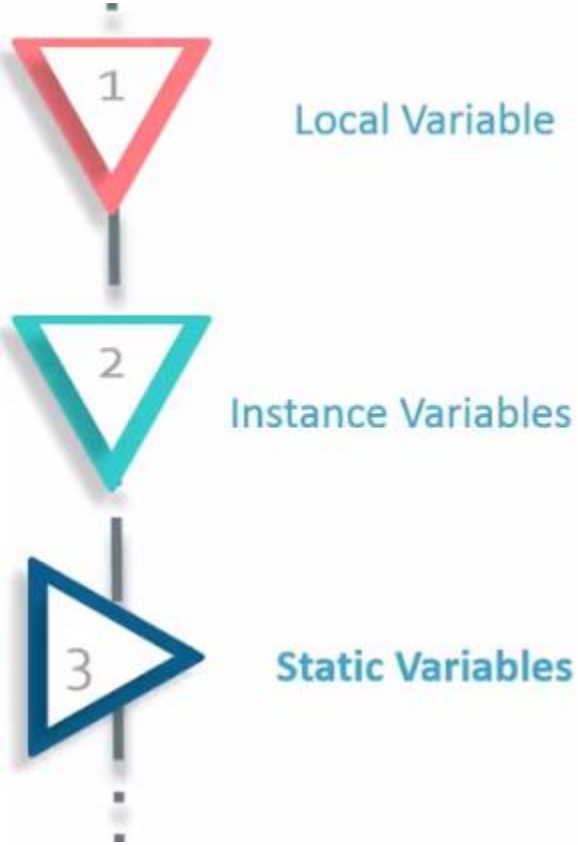
Lambda Expression: Variable Capture




```
public class LambdaStaticConsumerDemo {

    private String str = "Lambda Consumer";

    public void attach(LambdaStaticProducerDemo eventPr
        eventProd.listen(e -> {
            System.out.println(this.str);
        });
    }
}
```

Md. Mamun Hossain
B. Sc. (Engg.) & M. Sc. (Thesis) in CSE, SUST
Asst. Professor, Dept. of CSE , BAUST

Lambda Expression: Variable Capture




```
public class LambdaStaticConsumerDemo {
    private static String myStaticVar = "Edureka!";

    public void attach(LambdaStaticProducerDemo eventProd
        eventProd.listen(e -> {
            System.out.println(myStaticVar);
        });
    }
}
```

Md. Mamun Hossain
B. Sc. (Engg.) & M. Sc. (Thesis) in CSE, SUST
Asst. Professor, Dept. of CSE , BAUST

Lambda Expression: Variable Capture

Local Variable

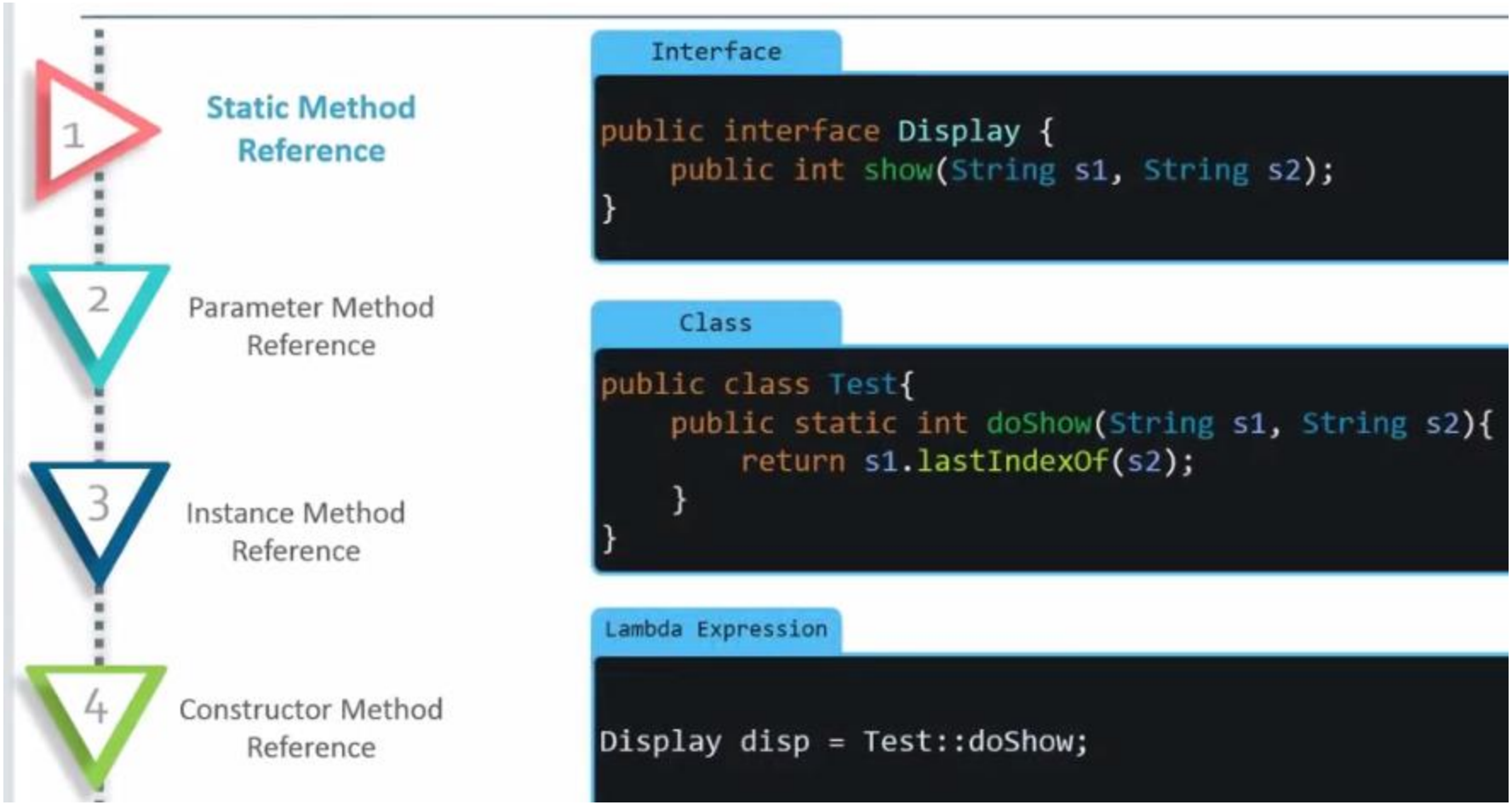
Instance Variables

Static Variables

```
public class LambdaApp {

    int instanceVar = 10;
    static int sVar = 100;

    public static void main(String[] args) {

        Cab cab = (source, destinition)->{
            int localVar = 1000;

            System.out.println("Local Var is: "+localVar);
            System.out.println("instanceVar is: "+instanceVar);
            System.out.println("Static Variable is: "+LambdaApp.

        };
```

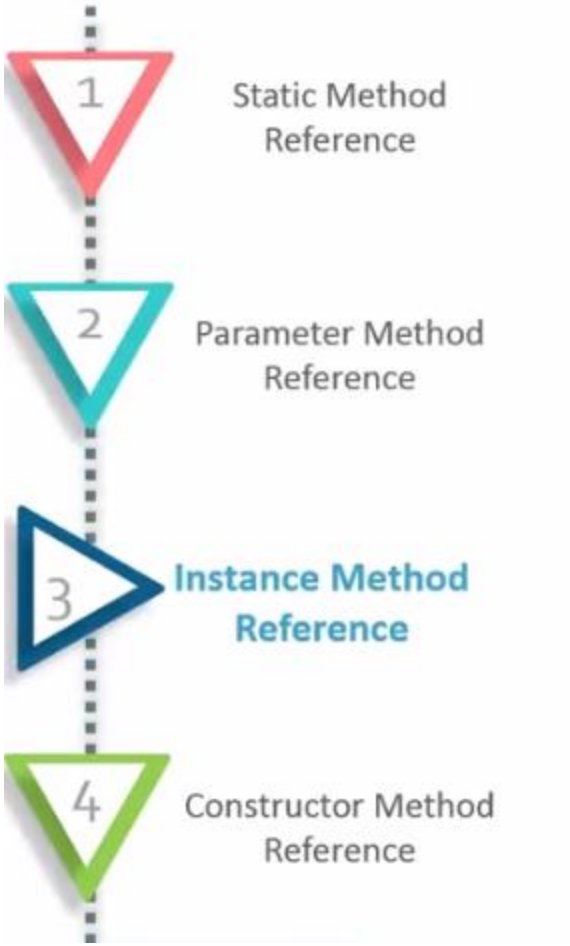
Md. Mamun Hossain
B. Sc. (Engg.) & M. Sc. (Thesis) in CSE, SUST
Asst. Professor, Dept. of CSE , BAUST

Lambda Expression: Method References



Md. Mamun Hossain
B. Sc. (Engg.) & M. Sc. (Thesis) in CSE, SUST
Asst. Professor, Dept. of CSE , BAUST

Lambda Expression: Method References



Interface

```
public interface Deserializer {
    public int deserialize(String v1);
}
```

Class

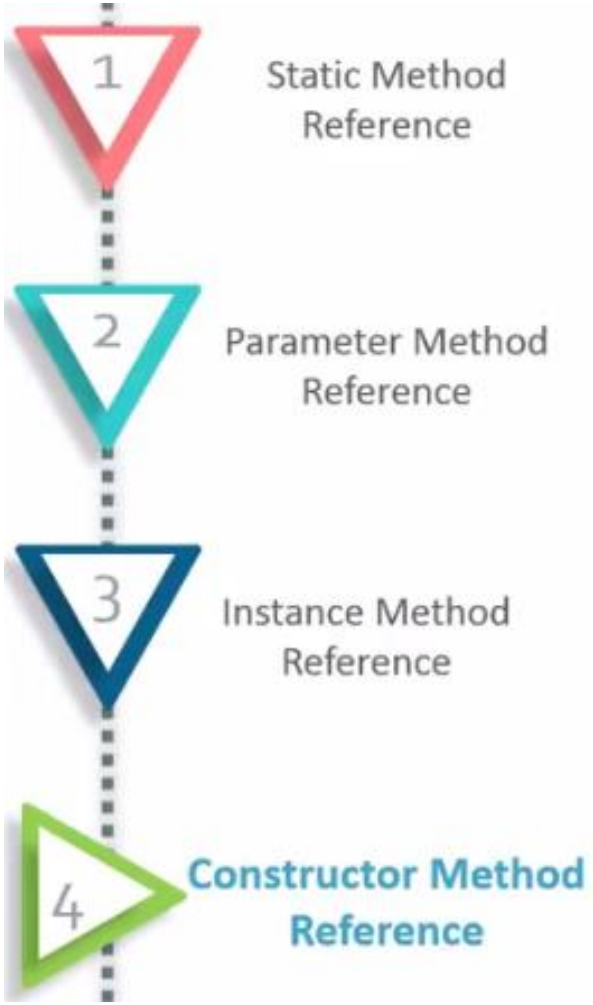
```
public class StringConverter {
    public int convertToInt(String v1){
        return Integer.valueOf(v1);
    }
}
```

Lambda Expression

```
StringConverter strConv = new StringConverter();

Deserializer deserializer = strConv::convertToInt;
```

Md. Mamun Hossain
B. Sc. (Engg.) & M. Sc. (Thesis) in CSE, SUST
Asst. Professor, Dept. of CSE , BAUST

Lambda Expression: Method References



Interface

```
public interface Factory {
    public String create(char[] val);
}
```

Lambda Expression

```
Factory fact = String::new;
```

Md. Mamun Hossain
B. Sc. (Engg.) & M. Sc. (Thesis) in CSE, SUST
Asst. Professor, Dept. of CSE , BAUST

Method References : Example

```
interface Calculator{
    void add(int num1, int num2);
}

class Calc{
    public static void addSomething(int num1, int num2){
        System.out.println(num1+" and "+num2+" addition is: "+(num1+num2));
    }
}

public class MethodReferencesApp {

    public static void main(String[] args) {

        Calc.addSomething(10, 20);

    }

}
```

```
10 and 20 addition is: 30
```

Md. Mamun Hossain
B. Sc. (Engg.) & M. Sc. (Thesis) in CSE, SUST
Asst. Professor, Dept. of CSE , BAUST

Method References : Static Method

```
interface Calculator{
    void add(int num1, int num2);
}

class Calc{
    public static void addSomething(int num1, int num2){
        System.out.println(num1+" and "+num2+" addition is: "+(num1+num2));
    }
}

public class MethodReferencesApp {

    public static void main(String[] args) {

        //Calc.addSomething(10, 20);

        //1. Reference to a Static Method
        Calculator cRef = Calc::addSomething; // Method Reference
        cRef.add(11, 14);



    }

}
```

```
10 and 20 addition is: 30
```

Md. Mamun Hossain
B. Sc. (Engg.) & M. Sc. (Thesis) in CSE, SUST
Asst. Professor, Dept. of CSE , BAUST

References to a Non Static/ instance Method

```
class Calc{
    public static void addSomething(int num1, int num2){
        System.out.println(num1+" and "+num2+" addition is: "+(num1+num2));
    }

    public void letsAdd(int num1, int num2){
        System.out.println(num1+" and "+num2+" adds to: "+(num1+num2));
    }
}

public class MethodReferencesApp {

    public static void main(String[] args) {

        //Calc.addSomething(10, 20);

        //1. Reference to a Static Method
        //Calculator cRef = Calc::addSomething; // Method Reference
        //cRef.add(11, 14);

        //2. Reference to a non static method or Instance Method
        // Object Construction Statement for Calc
        Calc calc = new Calc();
        Calculator cRef = calc::letsAdd;        // Method Reference
        cRef.add(12, 23);
```

```
10 and 20 addition is: 30
```

Md. Mamun Hossain
B. Sc. (Engg.) & M. Sc. (Thesis) in CSE, SUST
Asst. Professor, Dept. of CSE , BAUST

References to a Constructor

```
// Functional Interface
interface Messenger{
    Message getMessage(String msg);
}

class Message{
    Message(String msg){
        System.out.println(">> Message is: "+msg);
    }
}

public class MethodReferencesApp {

    public static void main(String[] args) {

        //Calc.addSomething(10, 20);

        //1. Reference to a Static Method
        //Calculator cRef = Calc::addSomething; // Method Reference
        //cRef.add(11, 14);

        //2. Reference to a non static method or Instance Method
        // Object Construction Statement for Calc
        //Calc calc = new Calc();
        //Calculator cRef = calc::letsAdd;      // Method Reference
        //cRef.add(12, 23);

        //3. Reference to a Constructor
        Messenger mRef = Message::new;          // Method Reference
        mRef.getMessage("Search the candle rather than cursing the darkness !!");
```

```
>> Message is: Search the candle rather than cursing the darkness !!
```

References to a Constructor

```
// Functional Interface
interface Calculator{
    void add(int num1, int num2);
}

class Calc{
    public static void addSomething(int num1, int num2){
        System.out.println(num1+" and "+num2+" addition is: "+(num1+num2));
    }

    public void letsAdd(int num1, int num2){
        System.out.println(num1+" and "+num2+" ad
    }
}

// Functional Interface
interface Messenger{
    Message getMessage(String msg);
}

class Message{
    Message(String msg){
        System.out.println(">> Message is: "+msg)
    }
}
```

```
public class MethodReferencesApp {

    public static void main(String[] args) {

        //Calc.addSomething(10, 20);

        //1. Reference to a Static Method
        //Calculator cRef = Calc::addSomething; // Method Reference
        //cRef.add(11, 14);

        //2. Reference to a non static method or Instance Method
        // Object Construction Statement for Calc
        //Calc calc = new Calc();
        //Calculator cRef = calc::letsAdd;      // Method Reference
        //cRef.add(12, 23);

        //3. Reference to a Constructor
        Messenger mRef = Message::new;          // Method Reference
        mRef.getMessage("Search the candle rather than cursing the da
```

```
>> Message is: Search the candle rather than cursing the darkne
```

Md. Mamun Hossain
B. Sc. (Engg.) & M. Sc. (Thesis) in CSE, SUST
Asst. Professor, Dept. of CSE , BAUST

Lambda Expression: Collection

- Lambda expressions are usually passed as parameters to a function:

Use a lamba expression in the `ArrayList`'s `forEach()` method to print every item in the list:

```
import java.util.ArrayList;

public class MyClass {
  public static void main(String[] args) {
    ArrayList<Integer> numbers = new ArrayList<Integer>();
    numbers.add(5);
    numbers.add(9);
    numbers.add(8);
    numbers.add(1);
    numbers.forEach( (n) -> { System.out.println(n); } );
  }
}
```

Md. Mamun Hossain
B. Sc. (Engg.) & M. Sc. (Thesis) in CSE, SUST
Asst. Professor, Dept. of CSE , BAUST

Lambda Expression: Collection

- Lambda expressions can be stored in variables if the variable's type is an interface which has only one method. The lambda expression should have the same number of parameters and the same return type as that method. Java has many of these kinds of interfaces built in, such as the Consumer interface (found in the java.util package) used by lists.

Use Java's `Consumer` interface to store a lambda expression in a variable:

```
import java.util.ArrayList;
import java.util.function.Consumer;

public class MyClass {
  public static void main(String[] args) {
    ArrayList<Integer> numbers = new ArrayList<Integer>();
    numbers.add(5);
    numbers.add(9);
    numbers.add(8);
    numbers.add(1);
    Consumer<Integer> method = (n) -> { System.out.println(n); };
    numbers.forEach( method );
  }
}
```

Lambda Expression: Collection

- To use a lambda expression in a method, the method should have a parameter with a single-method interface as its type. Calling the interface's method will run the lambda expression:

Create a method which takes a lambda expression as a parameter:

```
interface StringFunction {
  String run(String str);
}

public class MyClass {
  public static void main(String[] args) {
    StringFunction exclaim = (s) -> s + "!";
    StringFunction ask = (s) -> s + "?";
    printFormatted("Hello", exclaim);
    printFormatted("Hello", ask);
  }
  public static void printFormatted(String str, StringFunction format) {
    String result = format.run(str);
    System.out.println(result);
  }
}
```

Java RegEx

Regular Expression

a sequence of characters that forms a search pattern

Md. Mamun Hossain
B. Sc. (Engg.) & M. Sc. (Thesis) in CSE, SUST
Asst. Professor, Dept. of CSE , BAUST

Regular Expression

What is a Regular Expression?

- A regular expression is a sequence of characters that forms a search pattern. When you search for data in a text, you can use this search pattern to describe what you are searching for.

- A regular expression can be a single character, or a more complicated pattern.

- Regular expressions can be used to perform all types of text search and text replace operations.

Md. Mamun Hossain
B. Sc. (Engg.) & M. Sc. (Thesis) in CSE, SUST
Asst. Professor, Dept. of CSE , BAUST

Regular Expression

- Java does not have a built-in Regular Expression class, but we can import the java.util.regex package to work with regular expressions. The package includes the following classes:
  - **Pattern Class** - Defines a pattern (to be used in a search)
  - **Matcher Class** - Used to search for the pattern
  - **PatternSyntaxException Class** - Indicates syntax error in a regular expression pattern

Md. Mamun Hossain
B. Sc. (Engg.) & M. Sc. (Thesis) in CSE, SUST
Asst. Professor, Dept. of CSE , BAUST

Regular Expression : Example

- Find out if there are any occurrences of the word "w3schools" in a sentence:

```
import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class MyClass {
  public static void main(String[] args) {
    Pattern pattern = Pattern.compile("w3schools", Pattern.CASE_INSENSITIVE);
    Matcher matcher = pattern.matcher("Visit W3Schools!");
    boolean matchFound = matcher.find();
    if(matchFound) {
      System.out.println("Match found");
    } else {
      System.out.println("Match not found");
    }
  }
}
// Outputs Match found
```

Md. Mamun Hossain
B. Sc. (Engg.) & M. Sc. (Thesis) in CSE, SUST
Asst. Professor, Dept. of CSE , BAUST

Regular Expression : Explained

- **Example Explained**
- In this example, The word "w3schools" is being searched for in a sentence.
- First, the pattern is created using the Pattern.compile() method. The first parameter indicates which pattern is being searched for and the second parameter has a flag to indicates that the search should be case-insensitive. The second parameter is optional.
- The matcher() method is used to search for the pattern in a string. It returns a Matcher object which contains information about the search that was performed.
- The find() method returns true if the pattern was found in the string and false if it was not found.

Md. Mamun Hossain
B. Sc. (Engg.) & M. Sc. (Thesis) in CSE, SUST
Asst. Professor, Dept. of CSE , BAUST

Regular Expression : Flags

- **Flags**

Flags in the compile() method change how the search is performed. Here are a few of them:

- ***Pattern.CASE_INSENSITIVE*** - The case of letters will be ignored when performing a search.
- ***Pattern.LITERAL*** - Special characters in the pattern will not have any special meaning and will be treated as ordinary characters when performing a search.
- ***Pattern.UNICODE_CASE*** - Use it together with the CASE_INSENSITIVE flag to also ignore the case of letters outside of the English alphabet

Md. Mamun Hossain
B. Sc. (Engg.) & M. Sc. (Thesis) in CSE, SUST
Asst. Professor, Dept. of CSE , BAUST

Regular Expression : Patterns

- **Regular Expression Patterns**

The first parameter of the Pattern.compile() method is the pattern. It describes what is being searched for.

Brackets are used to find a range of characters:

Expression	Description
[abc]	Find one character from the options between the brackets
[^abc]	Find one character NOT between the brackets
[0-9]	Find one character from the range 0 to 9

Md. Mamun Hossain
B. Sc. (Engg.) & M. Sc. (Thesis) in CSE, SUST
Asst. Professor, Dept. of CSE , BAUST

Regular Expression : Metacharacters

- Metacharacters are characters with a special meaning:

Metacharacter	Description
|	Find a match for any one of the patterns separated by | as in: cat|dog|fish
.	Find just one instance of any character
^	Finds a match as the beginning of a string as in: ^Hello
$	Finds a match at the end of the string as in: World$
\d	Find a digit
\s	Find a whitespace character
\b	Find a match at the beginning of a word like this: \bWORD, or at the end of a word like this: WORD\b
\uxxxx	Find the Unicode character specified by the hexadecimal number xxxx

Md. Mamun Hossain
B. Sc. (Engg.) & M. Sc. (Thesis) in CSE, SUST
Asst. Professor, Dept. of CSE , BAUST

Regular Expression : Quantifiers

- Quantifiers define quantities:

Quantifier	Description
n+	Matches any string that contains at least one *n*
n*	Matches any string that contains zero or more occurrences of *n*
n?	Matches any string that contains zero or one occurrences of *n*
n{x}	Matches any string that contains a sequence of *X* *n*'s
n{x,y}	Matches any string that contains a sequence of X to Y *n*'s
n{x,}	Matches any string that contains a sequence of at least X *n*'s

Note: If your expression needs to search for one of the special characters you can use a backslash (\ ) to escape them. In Java, backslashes in strings need to be escaped themselves, so two backslashes are needed to escape special characters. For example, to search for one or more question marks you can use the following expression: "\\?"

Md. Mamun Hossain
B. Sc. (Engg.) & M. Sc. (Thesis) in CSE, SUST
Asst. Professor, Dept. of CSE , BAUST