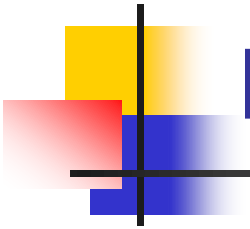
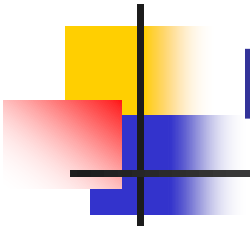
9. Lambda Expressions



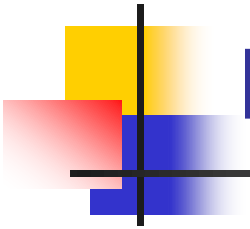
Lambda Expressions

- Lambda expression is a new and important feature of Java which was included in Java SE 8. It provides a clear and concise way to represent one method interface using an expression. It is very useful in collection library.



Lambda Expressions

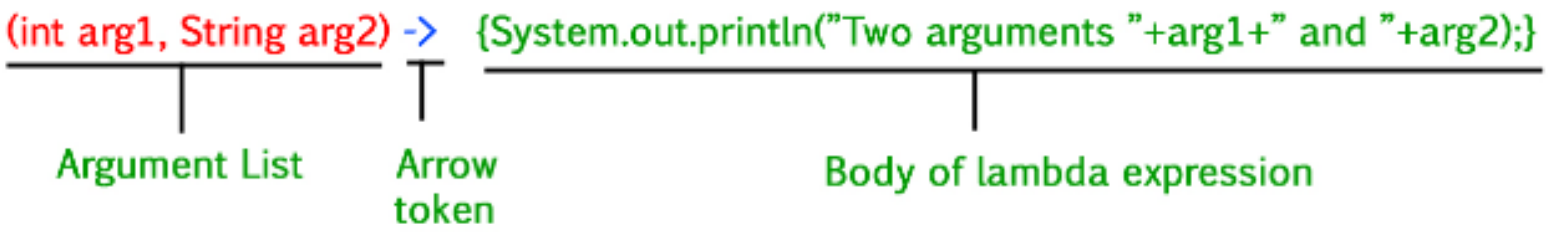
- One issue with anonymous classes is that if the implementation of your anonymous class is very simple, such as an interface that contains only one method, then the syntax of anonymous classes may seem unclear.
- Lambda expressions enable you to do this, to treat functionality as method argument, or code as data.

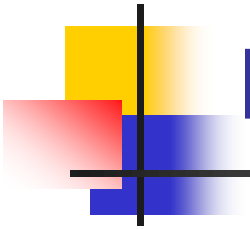


Lambda Expressions

- lambda expressions are added in Java 8 and provide below functionalities.
  - Enable to treat functionality as a method argument, or code as data.
  - A function that can be created without belonging to any class.
  - A lambda expression can be passed around as if it was an object and executed on demand.

Lambda Expressions





Lambda Expressions

- The body of a lambda expression can contain zero, one, or more statements.
- When there is a single statement curly brackets are not mandatory and the return type of the anonymous function is the same as that of the body expression.
- When there is more than one statement, then these must be enclosed in curly brackets (a code block) and the return type of the anonymous function is the same as the type of the value returned within the code block, or void if nothing is returned.

Lambda Expressions – Example #1

```
interface If1 {
    boolean fun(int n);
}

 // Lambda expression body
 If1 isEven = (n) -> (n % 2) == 0;

if (isEven.fun(21))
            // Display message to be printed
           System.out.println("21 is even");
else
           // Display message to be printed
           System.out.println("21 is odd");
```

Lambda Expressions – Example #2

```
// Interface
interface Func {
    // n is some natural number whose factorial is to be computed
    int fact(int n);
}


// Calling lambda function

// Print and display n the console
System.out.println("Factorial of 5 : " + f.fact(5));
```

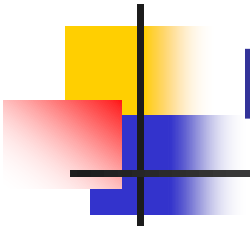
Lambda Expressions – Example #2

```
// Block lambda expression
Func f = (n) ->
{
        // Block body

        // Initially initializing with 1
        int res = 1;

        // iterating from 1 to the current number to find factorial by
         //multiplication
        for (int i = 1; i <= n; i++)
            res = i * res;
        return res;
};
```

Create Thread using Lambda Expressions



- Use of the Runnable Interface. As it is a Functional Interface, Lambda expressions can be used. The following steps are performed to achieve the task:
  - Create the Runnable interface reference and write the Lambda expression for the run() method.
  - Create a Thread class object passing the above-created reference of the Runnable interface since the start() method is defined in the Thread class its object needs to be created.
  - Invoke the start() method to run the thread.