Lambda Expression

what is lambda expression

- It can be defined as instance of functional interface
- it is also knowed as anonymous function
- Based on target variable that type of lambda expression is executed
- Earlier Annonymous class was very good for implementing logic interface or abstract classes method logic
- The lambda expression is shortcut of Annonymous inner class , but the only rule is theat ihe defined interface should be functional interface
- The rarget variable should be a function interface
- Then only this will execute or else it will give me error
- **This the syntax of FunctionalInterface**
  - FunctionInterface refVariable=(parameterList) -> { //logic };
  - we cannot create a lambda expression without creating a functional Interface
  - **always we must mention the target variable else the code will not** compile

what

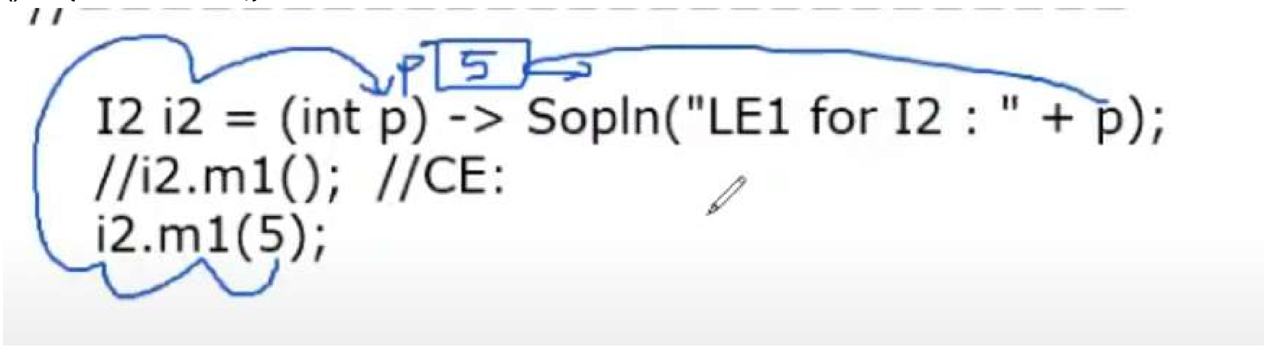
- **The proper Defination of lamba Expression**
  - It is an anonymous function ,It is an implementation of a functional Interface
  - it is an object of functional interface implimentation

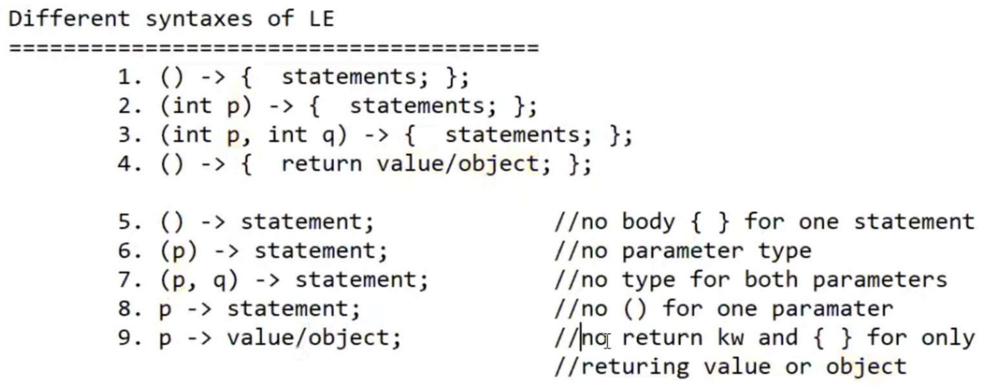
why

- It was introduced to implement functional programming paradigm
-

Analyze

- It has two parts
  - 1 part paraterlist (p1,p2,p3)
  - 2 part is Body
  - Those two parts are connected by lambda operator ->
  - It should end with ; because it is an expression
  - Lambda expression paramater and return type directly depends on functional interface method
  - we must not write return type in lambda expression but place return statement in lambda expression body
- **The four basic syntaxes of lambda expression are:**

- ()-> {// statements; };
- (int p) -> { statements; };
- (int p, int q) -> { statements; };
- () ->{return value;}
- 
  ```
  I2 i2 = (int p) -> Sopln("LE1 for I2 : " + p);
  //i2.m1();  //CE:
  i2.m1(5);
  ```

-
-
- ## Some rules for shortcut
  - **{}**
    - if you only write one statement we can remove {}
  - **PARAMETER IS OPTIONAL**
    - (int a,int b)->{ // statements; };
    - (a,b) -> { //statements; };
  - **For one parameter**
    - remove '(' and ')'
  - ### For return statements
    - For single statement body is optional
    - **return statement is not required**
    - For multiple statement
      - **body is required**
      - **return statement is also required**
        - if you write return statement body is required
      - 
        ```
        Different syntaxes of LE
        =======================================
                1. () -> {  statements; };
                2. (int p) -> {  statements; };
                3. (int p, int q) -> {  statements; };
                4. () -> {  return value/object; };

                5. () -> statement;             //no body { } for one statement
                6. (p) -> statement;            //no parameter type
                7. (p, q) -> statement;         //no type for both parameters
                8. p -> statement;              //no () for one paramater
                9. p -> value/object;           //no return kw and { } for only
                                                //returing value or object
        ```
      -

    •
•
•