**Lambdas and predicates**

|  |  |
| --- | --- |
| **Main** | **Employee** |
| **package** com.company;  **import** java.util.ArrayList; **import** java.util.List; **import** java.util.function.Predicate;  **public class** Main {   **public static void** main(String[] args) {  Employee snow = **new** Employee(**"Jon snow"**, 10);  Employee tygerrian = **new** Employee(**"Tygerrian"**, 5);  Employee ned = **new** Employee(**"Ned Stark"**, 25);  Employee lenisters = **new** Employee(**"Kingslayer"**, 35);   ArrayList<Employee> employees = **new** ArrayList<>();  employees.add(snow);  employees.add(tygerrian);  employees.add(ned);  employees.add(lenisters);   *//use of predicates  printEmployeesByAge*(employees, **"this are 20 up"**, employee -> employee.getExperience() > 20);  *printEmployeesByAge*(employees, **"this are less then 15"**, employee -> employee.getExperience() < 15);  }   **public static void** printEmployeesByAge(List<Employee> employees,  String agestring,  Predicate<Employee> agecondition){  System.***out***.println(agestring);  System.***out***.println(**"===================="**);  **for** (Employee emp : employees){  **if** (agecondition.test(emp)){  System.***out***.println(emp.getName());  }  }   } } | **package** com.company;  **class** Employee{  String **name**;  **int experience**;   *//constructor* **public** Employee(String name, **int** experience) {  **this**.**name** = name;  **this**.**experience** = experience;  }  *//getter and setter* **public** String getName() {  **return name**;  }  **public void** setName(String name) {  **this**.**name** = name;  }  **public int** getExperience() {  **return experience**;  }  **public void** setExperience(**int** experience) {  **this**.**experience** = experience;  } }  **this are 20 up**  **====================**  **Ned Stark**  **Kingslayer**  **this are less then 15**  **====================**  **Jon snow**  **Tygerrian** |

**Anonymous method as lambda expression**

|  |  |
| --- | --- |
| **package** com.company;  **import** java.util.ArrayList; **import** java.util.List; **import** java.util.function.Predicate;  **public class** Main {   **public static void** main(String[] args) {  Employee snow = **new** Employee(**"Jon snow"**, 10);  Employee tygerrian = **new** Employee(**"Tygerrian"**, 5);  Employee ned = **new** Employee(**"Ned Stark"**, 25);  Employee lenisters = **new** Employee(**"Kingslayer"**, 35);   ArrayList<Employee> employees = **new** ArrayList<>();  employees.add(snow);  employees.add(tygerrian);  employees.add(ned);  employees.add(lenisters);   *//use of predicates  printEmployeesByAge*(employees, **"this are 20 up"**,   employee -> employee.getExperience() > 20);  *printEmployeesByAge*(employees, **"this are less then 15"**,   employee -> employee.getExperience() < 15);   *//use of anonymous class in the place of  //PREDICATE in LAMBDA expression  printEmployeesByAge*(employees, **"this time cons are by anonymous method"**,   **new** Predicate<Employee>() {  @Override  **public boolean** test(Employee employee) {  **return** employee.getExperience() < 26;  }  });  }   **public static void** printEmployeesByAge(List<Employee> employees,  String agestring,  Predicate<Employee> agecondition){  System.***out***.println(agestring);  System.***out***.println(**"===================="**);  **for** (Employee emp : employees){  **if** (agecondition.test(emp)){  System.***out***.println(emp.getName());  }  }   } } | **this are 20 up**  **====================**  **Ned Stark**  **Kingslayer**  **this are less then 15**  **====================**  **Jon snow**  **Tygerrian**  **this time cons are by anonymous method**  **====================**  **Jon snow**  **Tygerrian**  **Ned Stark** |
| **package** com.company;  **class** Employee{  String **name**;  **int experience**;   *//constructor* **public** Employee(String name, **int** experience) {  **this**.**name** = name;  **this**.**experience** = experience;  }  *//getter and setter* **public** String getName() {  **return name**;  }  **public void** setName(String name) {  **this**.**name** = name;  }  **public int** getExperience() {  **return experience**;  }  **public void** setExperience(**int** experience) {  **this**.**experience** = experience;  } } |

**Single and duel consumer predicate**

|  |  |
| --- | --- |
| **package** com.company;  **import** java.util.ArrayList; **import** java.util.List; **import** java.util.function.IntPredicate; **import** java.util.function.Predicate;  **public class** Main {   **public static void** main(String[] args) {  *//single and double consumer predicate* IntPredicate x = i -> i > 15;  System.***out***.println(x.test(10)); *//single consumer* **int** a = 10;  System.***out***.println(x.test(10 + a)); *//dual consumer* } } | False  true |

**Chaining predicates**

|  |  |
| --- | --- |
| package com.company;  import java.util.ArrayList; import java.util.List; import java.util.function.IntPredicate; import java.util.function.Predicate;  public class Main {   public static void main(String[] args) {  *//single and double consumer predicate* IntPredicate greaterthan15 = i -> i > 15;  IntPredicate lessthan100 = i -> i < 100;   System.*out*.println(greaterthan15.and(lessthan100).test(60));  System.*out*.println(greaterthan15.and(lessthan100).test(6));  } } | true  false |

**Supplier**

|  |  |
| --- | --- |
| **package** com.company;  **import** java.util.ArrayList; **import** java.util.List; **import** java.util.Random; **import** java.util.function.IntPredicate; **import** java.util.function.Predicate; **import** java.util.function.Supplier;  **public class** Main {   **public static void** main(String[] args) {  Employee snow = **new** Employee(**"Jon snow"**, 10);  Employee tygerrian = **new** Employee(**"Tygerrian"**, 5);  Employee ned = **new** Employee(**"Ned Stark"**, 25);  Employee lenisters = **new** Employee(**"Kingslayer"**, 35);   ArrayList<Employee> employees = **new** ArrayList<>();  employees.add(snow);  employees.add(tygerrian);  employees.add(ned);  employees.add(lenisters);   Random random = **new** Random();  *//that cant be any od <a> Type* Supplier<Integer> sex = () -> random.nextInt(100);  **for** (**int** i = 0; i <= 10; i++){  System.***out***.println(sex.get());  }  } } | **31**  **94**  **28**  **40**  **68**  **40**  **3**  **75**  **72**  **73**  **53** |

1. **Consumer doesn’t return a VALUE.**
2. **Supplier doesn’t accept any PARAMETER.**
3. **Predicates only returns TRUE AND FALSE.**