

Lambda Ausdrücke

Ab Java 1.8 gab es eine neue Programmiererweiterung die sogenannten Lambda Ausdrücke.

Gesamtlernziel:

Lambda-Ausdrücke ermöglichen es, Funktionen als Parameter zu übergeben.
Bisher war das in Java nur indirekt über ein Objekt möglich

Um herauszufinden wofür ein Lambda Ausdruck unter anderem eingesetzt werden kann, sollen sie nun schrittweise den folgenden Code implementieren:

1. Schritt: Erstellen Sie eine Klasse Person

```
public class Person {  
    public enum Sex {MALE, FEMALE} // Enumeration: Aufzählung von symbolischen Konstanten  
  
    String name;  
    LocalDate birthday;  
    Sex gender;  
    String emailAddress;  
    int age;  
    .....  
}
```

Programmieren Sie die Klasse aus (Ergänzen Sie Zugriffsmodifizier, Konstruktor mit Parametern, set und get-Methoden und erstellen Sie eine Starter Klasse mit mindestens drei (4) Personen Objekten: s. Schritt 3)

2. Schritt: Implementieren Sie in der Starter Klasse eine Methode die nur Personen ausgibt die älter als ein gegebenes Alter sind.


```
public static void printPersonsOlderThan(List<Person> roster, int age) {  
    for (Person p : roster) {  
        if (p.getAge() >= age) {  
            System.out.println(p.toString());  
        }  
    }  
}
```

Die noch fehlende toString() Methode der Klasse Person kann man auch von Eclipse erzeugen lassen.

3. Schritt: Testen Sie Ihre Implementierung, indem Sie Tip 10 Jahre, Trick 11 Jahre, Track 9 Jahre, Daisy 39 Jahre und Donald 49 Jahre anlegen und in einer ArrayList mit Namen `pList` speichern. Danach Personen älter als 40 ausgeben lassen.

Teillernziel aus Schritt 1-3:

Wdh. der Generics, LocalDate Klasse , set und get Methoden, enum...

PR1	WA Lambda Ausdrücke (abgeändert durch Stk)	
-----	---	---

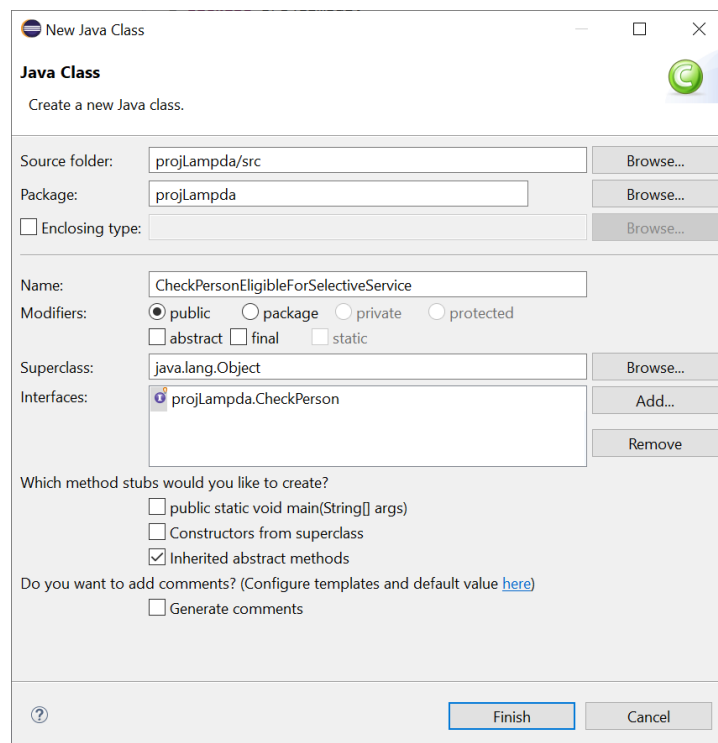
4. Schritt: Erstellen Sie in der Starter Klasse eine öffentliche, statische Methode `printPersons` mit folgendem Code:

```
public static void printPersons(List<Person> roster, CheckPerson tester) {
    for (Person p : roster) {
        if (tester.test(p)) {
            System.out.println(p.toString());
        }
    }
}
```

5. Schritt: Lassen Sie sich von eclipse helfen das folgende Interface zu erzeugen:


```
public interface CheckPerson {
    boolean test(Person p);
}
```

6. Schritt: Nun soll eine Prüfklasse erzeugt werden, die das Interface `CheckPerson` implementiert



```
public class CheckPersonEligibleForSelectiveService implements CheckPerson {

    @Override
    public boolean test(Person p) {
        return p.getGender() == Person.Sex.MALE
            && p.getAge() >= 10 && p.getAge() <= 40;
    }
}
```

PR1	WA Lambda Ausdrücke (abgeändert durch Stk)	
-----	---	---

- Welche Bedingungen werden hier abgeprüft?

7. Schritt: Erste Möglichkeit die Methode `printPersons` in der Starter Klasse aufzurufen:

```
printPersons(pList, new CheckPersonEligibleForSelectiveService());
```

Hier wird der Methode eine Instanz der Klasse `CheckPersonEligibleForSelectiveService` mitgegeben. Diese Klasse implementiert das verlangte Interface `CheckPerson` und damit auch die benötigte Methode `test()`.

8. Schritt: Nun eine zweite Möglichkeit, die Methode `printPersons` in der Starter Klasse aufzurufen:

```
printPersons( pList, new CheckPerson() {
    public boolean test(Person p) {
        return p.getGender() == Person.Sex.MALE && p.getAge() >= 10 &&
            p.getAge() <= 40; }
});
```

Hier wird eine Instanz (einer anonymen Klasse) im Aufruf erzeugt, die das verlangte Interface `CheckPerson` und die benötigte Methode `test()` implementiert. Dies kann noch einmal verkürzt werden.

9. Schritt: Nun kommt der **Lambda Ausdruck ins Spiel! In der Starter Klasse kann statt der anonymen Klasse direkt die erforderliche `test()` Methode in Form eines „Lambda Ausdrucks“ übergeben werden:**

```
printPersons(pList,
    (Person p) -> p.getGender() == Person.Sex.MALE && p.getAge() >= 18 &&
    p.getAge() <= 25);
```

Java Lambda Ausdruck allgemein

```
printPersons(roster,
    (Person p) -> p.getGender() == Person.Sex.MALE &&
    p.getAge() >= 18 && p.getAge() <= 25);
```

```
interface CheckPerson {
    boolean test(Person p);
}
```

Lambda-Operator „->“

Funktionales Interface

Der Compiler „weiß“, welches Interface von der Methode `printPersons` als zweiter Parameter erwartet wird.

Der Lambda Ausdruck muss dem Interface entsprechen, d.h. Parameter (hier `Person p`) und Rückgabtyp (hier `boolean`) müssen passen!