Sofia University
Department of Mathematics and Informatics

Course : OO Programming Java
Date: December 5, 2014
Student Name:

Lab No. 9

Използвайте изцяло средствата на Collections библиотеката за решаване на следните задачи

Задача 1.

а) Да се напишат Ламбда изрази от следните типове и да се дадат примери за тяхното изпълнение

```
Predicate<Integer>
BiPredicate<Computer, Computer>
Function<Computer, Double>
Consumer<Integer>
Supplier<String>
BinaryOperator<Integer >
```

b) Нека е даден метод

```
public void method(Function<Double, Double> function) {}
```

Дайте пример за изпълнение на този метод по един от следните начини:

1. **Като подавате за аргумент подходящо дефиниран Ламбда израз**
2. **Като подавате за аргумент подходящо дефиниран статичен метод**
3. **Като подавате за аргумент подходящо дефиниран нестатичен метод**

3) Да се определи стандартния тип на следните Ламбда изрази

```
(x) -> Math.cos(x)
(x) -> System.out.println(x);
()   -> System.out.println("Това е Ламбда");
(int x, int y) -> x > y;
(int x) -> x %2 == 0;
(int x, int y) -> x +  y;
```

4) Ако list **е** List<Integer>, **обяснете верижно Stream изпълнение на следната команда**

```
list.stream()
.filter(value -> value % 2 != 0)
.reduce(0, (x,y)-> x+ y);
```

5) Преобразувайте следния блок от сорс код към `Streams API`, за да се реализира външно итериране по елементите на масива `artists`

```
int totalMembers = 0;
for (Artist artist : artists) {
   Stream<Artist> members = artist.getMembers();
   totalMembers += members.count();
}
```

Тествайте получения сорс код като използвате примерния сорс код Lecture9bSampleCode.rar (в директория StreamsSamples) за `class Artist` и `class SampleData`.

Задача 2

Напишете Ламбда за изпълнението на следните задания:

a) Напишете Ламбда израз, който да се използва вместо следния анонимен клас:

```
new IntConsumer()
{
    public void accept(int value)
    {
       System.out.printf("%d ", value);
    }
}
```

b) Напишете функционален интерфейс съответен на следния Ламбда израз:

```
(String s) -> {return s.toUpperCase();}
```

С помощта на този интерфейс напишете анонимен клас, който е еквивалентен на дадения Ламбда израз

c) Напишете Ламбда израз без параметри , който връща String "Welcome to lambdas!„

d) Напишете Ламбда израз с два целочислени параметъра, който връща по- големия от тях

Задача 3

Напишете еквивалентно представяне на анонимен клас израз като използвате Ламбда за обработка на събитието

```
button.addActionListener(
 new ActionListener()
  {
   public void actionPerformed(ActionEvent event)
   {
     JOptionPane.showMessageDialog(ParentFrame.this,
                                   "JButton event handler");
   }
  }
);
```

Задача 4

Напишете клас `LambdaTest` и извършете следните действия

a) Напишете подходящи функционални интерфейси за всеки от следните Ламбда изрази:

- `(Integer x, Integer y) -> x + y`
- `(String s1, String s2) -> return String.format("%s, %s", s1, s2)`
- `(Double d1, Double d2) -> {`

```
                    Scanner input = new Scanner(System.in);
                    double d3 =   input.nextDouble();
                    return Math.max(d1,Math.max(d1,d2));
                };
```

b) Имплементирайте изразите в а) като анонимни класове с помощта на така предложените функционални интерфейси

c) **Напишете в `public static void main(String[] args)` метода на клас `LambdaTest`**

- изпълнете Ламбда изразите, посредством съответните им анонимни класове, а резултатът от изпълнението им да се изведе на стандартния изход
- команди за присвояване на изразите в а) на променливи от подходящо избран **стандартен** функционален интерфейс на Java. (използвайте параметър за тип)
- изпълнете Ламбда изразите, посредством така инициализираните променливи на **стандартен** функционален интерфейс, а резултатът от изпълнението им да се изведе на стандартния изход
- изпълнете Ламбда изразите, посредством така инициализираните променливи, а резултата от изпълнението им да се изведе на стандартния

d) Решете следната задача посредством Ламбда изрази и стандартни функционални интерфейси.

- **Напишете** ***class MySort***, който има статичен **метод** ***SortedSet<Integer> sort(int[] data, Comparator<Integer > sortOrder).***
  Нека този метод връща сортирано множество от различните елементи на масива ***data*** , при което функционалният интерефйс ***sortOrder*** задава на наредбата на елементите при сортиране. Използвайте клас TreeSet за реализация на сортирането
- Напишете ***class UseSort*** с метод ***main()*** за тестване на ***class MySort***.
  Дефинирайте един **масив** ***numbers*** **от цели числа** и **създайте обекти** от class ***MySort.***
  Дефинирайте два Ламбда израза от тип ***Comparator<Integer >*** съответно реферирани като ***upward*** и ***downward*** за наредба във възходящ ред и наредба в низходящ ред.
  Изведете на **стандартен изход** елементите на **зададения масив** и **същия масив след от изпълнението на метода *MySort*.sort() при сортирането на масива *numbers*** във **възходящ** и **низходящ** ред съответно с Ламбда изразите ***upward*** и do***w***nward.

Задача 5

Напишете в **class ArrayUtils** метод
static void filterNumbers(Predicate<Integer> condition, int[] array),
който **извежда на стандартен изхо**д елементите на масива **array,** за които Ламбда израза **condition** връща **true**

Създайте в **public static void main(String[] args)** метода на клас **ArrayUtils** масив **numbers** от **20 цели числа** с генератор за случайни числа в интервала [10- 50]. **Напишете** в този метод Ламбда изрази от тип **Predicate<Integer>**, които позволяват

a) да се изведат четните числа на масива **numbers**
b) да се изведат числата в интервала [30- 40]
c) да се изведат простите числа на масива **numbers**

Изпълнете метода **filterNumbers()** с всеки от тези Ламбда изрази

Задача 6

Дадени са **class** Salesperson и **class** LambdaDemo.

Инициализирайте predicate1 и predicate2 с Ламбда изрази, където

a) predicate1 връща true, когато getNumSales() на даден Salesperson е по-голямо от 1200
b) predicate2 връща true, когато getNumSales()на даден Salesperson е по-голямо от 900 и getSalary() е по- малко от 2500

Инициализирайте consumer1 и consumer2 с Ламбда изрази, където

a) consumer1 увеличава с 5% заплатата на даден Salesperson и извежда на стандартен изход всичките му данни
b) consumer2 увеличава с 2% заплатата на даден Salesperson и извежда в диалогов прозорец (JOptionPane.showMessageDialog()) всичките му данни

Допълнете метода applyBonus() с липсващите команди

Компилирайте **class** Salesperson и **class** LambdaDemo и **изпълнете** **class** LambdaDemo.

```
public class Salesperson
{
   private String name;
   private double salary;
   private int numsales;

  public   Salesperson(String name, int salary, int numsales)
   {
      this.name = name;
      this.salary = salary;
      this.numsales = numsales;
   }

  public   void addBonus(double amount)
   {
      salary += amount;
   }

  public   int getNumSales()
   {
      return numsales;
   }
  public   int getSalary()
   {
      return salary;
   }
   @Override
   public String toString()
   {
      return String.format("name: %s, salary: %.2f numsales: %d ",
                           name, salary, numsales);
}
public class LambdaDemo
{
   public static void main(String[] args)
   {
      Predicate<Salesperson> predicate1; // да се инициализира
      Predicate<Salesperson> predicate2; // да се инициализира
      Consumer<Salesperson> consumer1; // да се инициализира
      Consumer<Salesperson> consumer2; // да се инициализира

      Salesperson[] salespersons =
      {
         new Salesperson("John Doe", 2000, 949),
         new Salesperson("Jane Doe", 3900, 1500)
      };

      for (Salesperson salesperson: salespersons)
      {
         applyBonus(salesperson,
                    predicate1,
                    consumer1);
         System.out.println(salesperson);
      }
      for (Salesperson salesperson: salespersons)
      {
         applyBonus(salesperson,
```

```
                    prediacte2,
                    consumer2);
            System.out.println(salesperson);
        }

    }

    public static void applyBonus(Salesperson salesperson,
                                  Predicate<Salesperson> predicate,
                                  Consumer<Salesperson> consumer)
    {
        // Напишете if  команда, където използвайте predicate
        //    за да определите дали да изпълните consumer
        // Изпълнете consumer, когато условието на if  командата е изпълнено
    }
}
```