Module 5 - Java Collections, Generics

Advanced Java Certification Training

Akram M'Tir

1. Write the subtract(...), divide(...) methods in module5.generics. ArithmeticOperations class.

ArithmeticOperations.java | FriendshipCriteria.java | Friend.java

```java
package module5;

import java.util.Arrays;
import java.util.List;

public class ArithmeticOperations {

    public static <T extends Number> Number add(T t1, T t2) {
        return (t1.doubleValue() + t2.doubleValue() );
    }

    public static <T extends Number> Number subtract(T t1, T t2) {
        return (t1.doubleValue() - t2.doubleValue() );
    }

    public static <T extends Number> Number divide(T t1, T t2) {
        return (t1.doubleValue() / t2.doubleValue() );
    }

    public static <T extends Number> Number add( List<? extends Number> list) {
        double d =0;
        for(int i=0; i<list.size();i++) {
            d += list.get(i).doubleValue();
        }
        return new Double(d);
    }
```

Problems | Javadoc | Declaration | Console | Debug

```
<terminated> ArithmeticOperations [Java Application] /usr/lib/jvm/java-1.8.0-openjdk-1.8.0.144-0.b01.el7.
Add with generic methood: 77.0
Float Numbers: 12.56, 3.6778
Add with generic methood: 16.237800359725952
Subtract with generic methood: 8.882200479507446
Divide with generic methood: 3.4150852749330327
```

2. Write a generic method to swap positions in any kind of list [Solution.module5.generics.GenericUtils].

a) Method signature: public static <T> T[] swap(T [] list, int firstPos, int secondPos)

b) Throw appropriate exceptions if indexes are out of bounds.

ArithmeticOperations.java | FriendshipCriteria.java | Friend.java

```java
    public static <T> T[] swap(T[] list, int firstPost, int secondPos)  {
        // Check the length of the array
        if(firstPost >= list.length && firstPost >= list.length ) {
            throw new IndexOutOfBoundsException("Please check firstPost and
        }
        T temp = list[firstPost];
        list[firstPost] = list[secondPos];
        list[secondPos] = temp;
        return list;
    }

    public static void dumpList(List<?> list) {
        System.out.println("List dump with unbounded wildcard: ");
        for(int i =0; i<list.size(); i++)
            System.out.println(list.get(i));
    }

    public static <T>  void dumpList(T[] arr) {
        System.out.println("List dump with unbounded wildcard: ");
        for(int i =0; i<arr.length; i++)
            System.out.println(arr[i]);
    }

    public static void main(String[] args) {
        Integer i1 = new Integer(34);
        Integer i2 = new Integer(43);
```

Problems | Javadoc | Declaration | Console | Debug

```
<terminated> ArithmeticOperations [Java Application] /usr/lib/jvm/java-1.8.0-openjdk-1.8.0.144-0.b
Add with generic methood: 77.0
Float Numbers: 12.56, 3.6778
Add with generic methood: 16.237800359725952
Subtract with generic methood: 8.882200479507446
Divide with generic methood: 3.4150852749330327
Before Swaping [1, 2, 3, 4, 5]
After Swaping [5, 2, 3, 4, 1]
```

3. Write a generic class FriendshipCriteria with attributes T & S.

a. T & S should implement java.lang.Comparable.

b. Write a programme to find friends (FriendFinder) when T = lava.lang.String (which is the name) and S = java.lang.Integer (which is the age).

c. Write a programme to find friends (FriendFinder) when T = lava.lang.String (which is the name) and S = java.lang.String (which is the location).

d. Try T & S with user defined classes.

i. [Solution: module5.generics.FriendFinder, module5.generics.FriendshipCriteria]

ArithmeticOperations.java | FriendshipCriteria.java | Friend.java

```java
package module5;

public class Friend implements Comparable<Friend>{
    private String name;
    private Integer age;
    private String location;
    public Friend(String name, Integer age, String location) {
        this.name = name;
        this.age = age;
        this.location = location;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public Integer getAge() {
        return age;
    }
    public void setAge(int age) {
        this.age = age;
    }
    public String getLocation() {
        return location;
    }
    public void setLocation(String location) {
        this.location = location;
    }
    @Override
    public String toString() {
        return "Friend [name=" + name + ", age=" + age + ", location=" + location + "]";
    }
//  Integer and String implement both the comparable interface.
    public  <T extends String, S extends Integer> boolean friendFinder(T t, S s){
        if ( (t.equals(this.getName()) && s == this.getAge())    )
            return true;
        else
            return false;
```

```java
    @Override
    public String toString() {
        return "Friend [name=" + name + ", age=" + age + ", location=" + location + "]";
    }
//  Integer and String implement both the comparable interface.
    public  <T extends String, S extends Integer> boolean friendFinder(T t, S s){
        if ( (t.equals(this.getName()) && s == this.getAge())    )
            return true;
        else
            return false;
    }

    public  <T extends String, S extends String> boolean friendFinder(T t, S s){
        if ( (t.equals(this.getName()) && s.equals(this.getLocation() )    ) )
            return true;
        else
            return false;
    }

@Override
public int compareTo(Friend fr) {
    // If friend object are equal they should have the same name, age and location
    if ( (fr.getName().equals(this.getName()) && fr.getAge() == this.getAge()  &&  fr.getLocation() == this.getLocation() )    )
        return 0;
    else if ( this.getAge() > fr.getAge()   ) // compare age to sort
        return 1;
    else
        return -1;
}


}
```

ArithmeticOperations.java | FriendshipCriteria.java | Friend.java

```java
package module5;

import java.util.ArrayList;
import java.util.Collections;
import java.util.List;

public class FriendshipCriteria {
    public static void main(String[] args) {
        // Criteria
        String name = "Bob";
        Integer age = 25;
        String location = "Paris";
        Friend f = new Friend(name, age, location );
        // List of Friend Object
        List<Friend> list = new ArrayList<Friend>();
        list.add(new Friend("Bob", 25, "Paris"));
        list.add(new Friend("Tom", 35, "London"));
        list.add(new Friend("Alice", 45, "Madrid"));
        list.add(new Friend("Peter", 50, "Berlin"));
        list.add(new Friend("Bob", 25, "New York"));
        list.add(new Friend("Alex", 35, "London"));
        list.add(new Friend("David", 45, "Madrid"));
        list.add(new Friend("Bob", 55, "Paris"));
        list.add(new Friend("Sandy", 50, "Berlin"));
        // Printing the list of friends
        System.out.println("list of friend");
        for(Friend fr : list) {
            System.out.println("Another friend : " + fr);
        }
        // Sorting the list
        Collections.sort(list);
        // Printing the list sorted by age
        System.out.println("");
        System.out.println("list of friend sorted by age: ");
        for(Friend fr : list) {
            System.out.println("Another friend : " + fr);
        }
```

```java
        // Sorting the list
        Collections.sort(list);
        // Printing the list sorted by age
        System.out.println("");
        System.out.println("list of friend sorted by age: ");
        for(Friend fr : list) {
            System.out.println("Another friend : " + fr);
        }

        System.out.println("------");  System.out.println("Friends based on name and age: ");
        for(Friend fr : list) {

        if ( fr.friendFinder(name, age) ) // if ( fr.equals() )
                System.out.println("Found a friend for you : " + fr);
        }

        System.out.println("------");  System.out.println("Friends based on name and location: ");
        for(Friend fr : list) {

        if ( fr.friendFinder(name, location) ) // if ( fr.equals() )
                System.out.println("Found a friend for you : " + fr);
        }

    }
}
```

Problems @ Javadoc Declaration Console Debug

```
<terminated> FriendshipCriteria [Java Application] /usr/lib/jvm/java-1.8.0-openjdk-1.8.0.
list of friend
Another friend : Friend [name=Bob, age=25, location=Paris]
Another friend : Friend [name=Tom, age=35, location=London]
Another friend : Friend [name=Alice, age=45, location=Madrid]
Another friend : Friend [name=Peter, age=50, location=Berlin]
Another friend : Friend [name=Bob, age=25, location=New York]
Another friend : Friend [name=Alex, age=35, location=London]
Another friend : Friend [name=David, age=45, location=Madrid]
Another friend : Friend [name=Bob, age=55, location=Paris]
Another friend : Friend [name=Sandy, age=50, location=Berlin]

list of friend sorted by age:
Another friend : Friend [name=Bob, age=25, location=New York]
Another friend : Friend [name=Bob, age=25, location=Paris]
Another friend : Friend [name=Alex, age=35, location=London]
Another friend : Friend [name=Tom, age=35, location=London]
Another friend : Friend [name=David, age=45, location=Madrid]
Another friend : Friend [name=Alice, age=45, location=Madrid]
Another friend : Friend [name=Sandy, age=50, location=Berlin]
Another friend : Friend [name=Peter, age=50, location=Berlin]
Another friend : Friend [name=Bob, age=55, location=Paris]
------
Friends based on name and age:
Found a friend for you : Friend [name=Bob, age=25, location=New York]
Found a friend for you : Friend [name=Bob, age=25, location=Paris]
------
Friends based on name and location:
Found a friend for you : Friend [name=Bob, age=25, location=Paris]
Found a friend for you : Friend [name=Bob, age=55, location=Paris]
```