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

```
  package com.capgemini.generics;

import java.util.List;
import java.util.Vector;

/**
 * Supports arithmetic operations of +, -, *, / on any type that are subclasses
 * of {@link java.lang.Number}.
 * @author pchandra
 *
 */
public class ArithmeticOperations
{
	/**
	 * Generic method to add 2 numbers.
	 * @param t1
	 * @param t2
	 * @return
	 */
	public static <T extends Number> Number add(T t1, T t2)
	{ return (t1.doubleValue() + t2.doubleValue()); }

	/**
	 * Demostrates Upper Bounded Wildcards.
	 * Generic method to add numbers from a list.
	 * @param t1
	 * @param t2
	 * @return
	 */
	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);
	}

	/**
	 * Demostrates Unbounded Wildcards.
	 * Generic method to dump list data to console.
	 * @param t1
	 * @param t2
	 * @return
	 */
	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 void main(String[] args)
```

```
	{
		// Adding 2 integers
		Integer i1 = new Integer(34), i2 = new Integer(43);
		System.out.println("Add with generic method: " +
ArithmeticOperations.add(i1, i2));
		Float f1 = new Float(12.56), f2 = new Float(3.6778);
		System.out.println("Add with generic method: " +
ArithmeticOperations.add(f1, f2));

		// Adding 2 integers through a list
		Vector<Number> l = new Vector<Number>();
		l.add(new Integer(34));
		l.add(new Integer(43));
		System.out.println("Add with upper bounded wildcard: " +
ArithmeticOperations.add(l));

		// Dumping the list to the console.
		ArithmeticOperations.dumpList(l);
	}
}
```

2.Write a generic method to swap positions in any kind of list. [com.capgemini.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.

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.[com.capgemini.generics.FriendFinder, com.capgemini.generics..FriendshipCriteria]