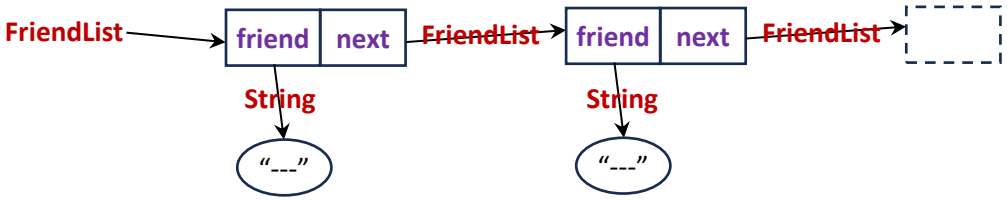
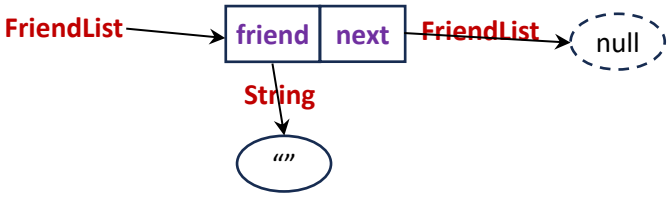
INT101 In-class exercises on October 18-19, 2023 (**Problem and Solution**)

1. Create a project named `int101exercise01`.
2. Create a utility class named `Tool` in `work01` package.
   2.1. Make the class a **`public final`** class.
   2.2. Create a **`private`** empty constructor that does nothing.
   2.3. Create a **`public static`** `median` method that receives three **`double`** parameters named `d0`, `d1`, and `d2`; and returns a **`double`** which is the median of the three parameters.
   2.4. Create a **`public static`** `compute` method that receives three parameters which are `d0` (**`double`**), `d1` (**`double`**), and `operation` (**`String`**). This method returns a **`double`** which is the result of the following calculation.
      2.4.1. `d0 + d1` if the `operation` is "`sum`", "`add`", or "`plus`",
      2.4.2. `d0 - d1` if the `operation` is "`difference`", "`subtract`", or "`minus`",
      2.4.3. `d0 * d1` if the `operation` is "`product`", "`multiply`", or "`times`",
      2.4.4. `d0 / d1` if the `operation` is "`division`" or "`by`".
      2.4.5. Otherwise, it throws a `RuntimeException` with a message "`Invalid Operation`".
      2.4.6. Use a switch-case expression with arrow cases.
   2.5. Create a **`public static`** `digitProduct` method that receives an **`int`** parameter named `number` and returns an **`int`** which is the product of all digits (except `0`) in the `number` parameter. However, this method will return `-1` if the `number` parameter is a negative number and return `1` if the `number` parameter is `0`.
   E.g., `digitProduct(2090075)` will return `630` which is the result of `2 * 9 * 7 * 5`.
3. Create a `FriendList` class in `work02` package.
   3.1. Make this class a **`public`** class.
   3.2. Create a **`private final`** `friend` field of type **`String`**.
   3.3. Create a **`private`** `next` field of type `FriendList`.



   3.4. Create a **`private`** constructor that receives a `friend` parameter (**`String`**) to set the `friend` field.
   3.5. Create a **`public static`** `newList` method that receives no parameters but returns a newly-created `FriendList` with an empty string as its `friend`.



   3.6. Create a public `addFriend` method that receives a `friend` parameter (**`String`**) and returns a **`boolean`**. This method returns **`false`** if the `friend` parameter is **`null`** or a blank string or equals to any `friend` field of any `FriendList` in the `next` chained list. Otherwise, it creates a new `FriendList` object with the `friend` field equals to the `friend` parameter and put this newly-created `FriendList` object as the `next` field of the last `FriendList` in the chained list.
   3.7. Create an overriding **`public`** `toString` method that receives no parameters and returns a **`String`** containing the values of all the `friend` fields of all `FriendList` object in the chained list separated by commas (`", "`). Use **`StringBuilder`** to build the output **`String`**. Return an empty string if there are no friends on the list.

(Please continue the next page.)

4. Create an **Exercise** class named in a **mainprogram** package to test the **Tool** class and the **FriendList** class.
   4.1. create a **static testMedian** method that receives no parameters to test the **median** method in the **Tool** class. You must test the method with many test cases to make sure that you cover all possible outcomes.
   4.2. create a **static testCompute** method that receives no parameters to test the **compute** method in the **Tool** class. You must test the method with many test cases to make sure that you cover all possible outcomes.
   4.3. create a **static testDigitProduct** method that receives no parameters to test the **digitProduct** method in the **Tool** class. You must test the method with many test cases to make sure that you cover all possible outcomes.
   4.4. Create a **static testFriendList** method that receives no parameters to test all the methods in the **FriendList** class. You must test it with many test cases to make sure that you cover all possible outcomes.
   4.5. create a **public static void main(String[] args)** method to call all the above test methods in this class.

----- END OF EXERCISE ----------

```
//SOLUTION PART 1 OF 3 ----------------------------------------
//Path: work01/Tool.java

package work01; //2.

public final class Tool { //2.,2.1
   private Tool() {} //2.2
   public static double median(double d0, double d1, double d2) { //2.3
      if (d0<d1) {
         if (d1<d2) return d1; // d0<d1<d2
         if (d0<d2) return d2; // d0<d2<=d1
         return d0; // d2<=d0<d1
      }
      if (d0<d2) return d0; // d1<=d0<d2
      if (d1<d2) return d2; // d1<d2<=d0
      return d1; // d2<=d1<=d0
   }
   public static double compute(double d0, double d1, String operation) { //2.4
      return
         switch(operation) {
            case "sum","add","plus" -> d0+d1;
            case "difference","subtract","minus" -> d0-d1;
            case "product","multiply","times" -> d0*d1;
            case "division","by" -> d0/d1;
            default -> throw new RuntimeException("Invalid Operation");
         };
   }
   public static int digitProduct(int number) { //2.5
      if (number < 0) return -1;
      if (number == 0) return 1;
      int product = 1;
      while (number > 1) {
         int mod = number % 10;
         if (mod > 1) product *= mod;
         number /= 10;
      }
      return product;
   }
}
```

(Please continue the next page.)

```java
//SOLUTION PART 2 OF 3 ---------------------------------------
//Path: work02/FriendList.java

package work02; //3.

public class FriendList { //3.,3.1
   private final String friend; //3.2
   private FriendList next; //3.3
   private FriendList(String friend) { //3.4
      this.friend = friend;
   }
   public static FriendList newList() { //3.5
      return new FriendList("");
   }
   public boolean addFriend(String friend) { //3.6
      if (friend == null || friend.isBlank()) return false;
      var current = this;
      while (current.next != null) {
         current = current.next;
         if (current.friend.equals(friend)) return false;
      }
      current.next = new FriendList(friend);
      return true;
   }
   @Override
   public String toString() { //3.7
      var current = this.next;
      if (current == null) return "";
      var s = new StringBuilder();
      s.append(current.friend);
      while ((current=current.next) != null) {
         s.append(", ").append(current.friend);
      }
      return s.toString();
   }
}

//SOLUTION PART 3 OF 3 ---------------------------------------
//Path: mainprogram/Exercise.java

package mainprogram; //4.

import work01.Tool; //4.1,4.2,4.3
import work02.FriendList; //4.4

public class Exercise { //4.

    public static void main(String[] args) { //4.5
       testMedian();
       testCompute();
       testDigitProduct();
       testFriendList();
    }
```

// (Please continue the next page.)

```
    static void testMedian() { //4.1
        System.out.println("## test Tool.median() ##");
        double x=1.0,y=2.0,z=3.0;
        System.out.format("median of %3.1f,%3.1f,%3.1f = %3.1f%n",
            x,y,z,Tool.median(x,y,z));
        System.out.format("median of %3.1f,%3.1f,%3.1f = %3.1f%n",
            x,z,y,Tool.median(x,z,y));
        System.out.format("median of %3.1f,%3.1f,%3.1f = %3.1f%n",
            y,x,z,Tool.median(y,x,z));
        System.out.format("median of %3.1f,%3.1f,%3.1f = %3.1f%n",
            y,z,x,Tool.median(y,z,x));
        System.out.format("median of %3.1f,%3.1f,%3.1f = %3.1f%n",
            z,x,y,Tool.median(z,x,y));
        System.out.format("median of %3.1f,%3.1f,%3.1f = %3.1f%n",
            z,y,x,Tool.median(z,y,x));
        System.out.format("median of %3.1f,%3.1f,%3.1f = %3.1f%n",
            x,y,y,Tool.median(x,y,y));
        System.out.format("median of %3.1f,%3.1f,%3.1f = %3.1f%n",
            y,x,y,Tool.median(y,x,y));
        System.out.format("median of %3.1f,%3.1f,%3.1f = %3.1f%n",
            y,y,x,Tool.median(y,y,x));
        System.out.format("median of %3.1f,%3.1f,%3.1f = %3.1f%n",
            x,x,y,Tool.median(x,x,y));
        System.out.format("median of %3.1f,%3.1f,%3.1f = %3.1f%n",
            x,y,x,Tool.median(x,y,x));
        System.out.format("median of %3.1f,%3.1f,%3.1f = %3.1f%n",
            y,x,x,Tool.median(y,x,x));
        System.out.format("median of %3.1f,%3.1f,%3.1f = %3.1f%n",
            z,z,z,Tool.median(z,z,z));
    }

    static void testCompute() { //4.2
        System.out.println("## test Tool.compute() ##");
        double d0=9.0,d1=2.0;
        String op = "sum";
        System.out.format("%3.1f,%3.1f,%s = %3.1f%n",d0,d1,op,Tool.compute(d0,d1,op));
        op = "add";
        System.out.format("%3.1f,%3.1f,%s = %3.1f%n",d0,d1,op,Tool.compute(d0,d1,op));
        op = "plus";
        System.out.format("%3.1f,%3.1f,%s = %3.1f%n",d0,d1,op,Tool.compute(d0,d1,op));
        op = "difference";
        System.out.format("%3.1f,%3.1f,%s = %3.1f%n",d0,d1,op,Tool.compute(d0,d1,op));
        op = "subtract";
        System.out.format("%3.1f,%3.1f,%s = %3.1f%n",d0,d1,op,Tool.compute(d0,d1,op));
        op = "minus";
        System.out.format("%3.1f,%3.1f,%s = %3.1f%n",d0,d1,op,Tool.compute(d0,d1,op));
        op = "product";
        System.out.format("%3.1f,%3.1f,%s = %3.1f%n",d0,d1,op,Tool.compute(d0,d1,op));
        op = "multiply";
        System.out.format("%3.1f,%3.1f,%s = %3.1f%n",d0,d1,op,Tool.compute(d0,d1,op));
        op = "times";
        System.out.format("%3.1f,%3.1f,%s = %3.1f%n",d0,d1,op,Tool.compute(d0,d1,op));
        op = "division";
        System.out.format("%3.1f,%3.1f,%s = %3.1f%n",d0,d1,op,Tool.compute(d0,d1,op));
        op = "by";
        System.out.format("%3.1f,%3.1f,%s = %3.1f%n",d0,d1,op,Tool.compute(d0,d1,op));
```

// (Please continue the next page.)

```
        d1=0.0; // divided by zero
        System.out.format("%3.1f,%3.1f,%s = %3.1f%n",d0,d1,op,Tool.compute(d0,d1,op));
        d0=-3.0; // a negative number divided by zero
        System.out.format("%3.1f,%3.1f,%s = %3.1f%n",d0,d1,op,Tool.compute(d0,d1,op));
        d0=0.0; // zero divided by zero
        System.out.format("%3.1f,%3.1f,%s = %3.1f%n",d0,d1,op,Tool.compute(d0,d1,op));

        op ="power"; // an invalid operation
        try {
           System.out.format("%3.1f,%3.1f,%s = %3.1f%n",
              d0,d1,op,Tool.compute(d0,d1,op));
        } catch (Exception e) {
           System.out.format("Operation %s with %3.1f, %3.1f: %s%n",op,d0,d1,e);
        }
    }

    static void testDigitProduct() { //4.3
        System.out.println("## test Tool.digitProduct() ##");
        int n = 20900751;
        System.out.format("product of all digits in %d ... %d%n",n,Tool.digitProduct(n));
        n = 165310;
        System.out.format("product of all digits in %d ... %d%n",n,Tool.digitProduct(n));
        n = -234;
        System.out.format("product of all digits in %d ... %d%n",n,Tool.digitProduct(n));
        n = 0;
        System.out.format("product of all digits in %d ... %d%n",n,Tool.digitProduct(n));
        n = 1;
        System.out.format("product of all digits in %d ... %d%n",n,Tool.digitProduct(n));
    }

    static void testFriendList() { //4.4
        System.out.println("## test FriendList ##");
        FriendList f = FriendList.newList();
        System.out.format("f: [%s]%n", f);
        String name = null;
        System.out.format("f.addFriend(%s): %b --> [%s]%n",name,f.addFriend(name),f);
        name = "first";
        System.out.format("f.addFriend(%s): %b --> [%s]%n",name,f.addFriend(name),f);
        name = "second";
        System.out.format("f.addFriend(%s): %b --> [%s]%n",name,f.addFriend(name),f);
        name = "third";
        System.out.format("f.addFriend(%s): %b --> [%s]%n",name,f.addFriend(name),f);
        name = "second"; // a duplicate name
        System.out.format("f.addFriend(%s): %b --> [%s]%n",name,f.addFriend(name),f);
        name = "first"; // a duplicate name
        System.out.format("f.addFriend(%s): %b --> [%s]%n",name,f.addFriend(name),f);
        name = "forth";
        System.out.format("f.addFriend(%s): %b --> [%s]%n",name,f.addFriend(name),f);
        name = "  "; // a blank name
        System.out.format("f.addFriend(%s): %b --> [%s]%n",name,f.addFriend(name),f);
        name = ""; // an empty name
        System.out.format("f.addFriend(%s): %b --> [%s]%n",name,f.addFriend(name),f);
        name = "fifth";
        System.out.format("f.addFriend(%s): %b --> [%s]%n",name,f.addFriend(name),f);
    }
}

//END OF SOLUTION ---------------------------------------
```