Final 40%

Make the Project and named it FinalExam , for each question create the Pakage and name them Q1,Q2,Q3,Q4

Question 1 : 10%

You are assigned to create the Hash Table with only array . You have to find the maximum number of Collisions for this hashing and say in which index will be landed.
Then you need to print the hash table array.
The size of the table is come from users.(Hint :Make sure the size of the array should be prime but user can put any number)

Example 1 :

```
Size of hash table?
7
Give me the numbers?
3 4 5 1 9 1 4
MAX COLLISIONS: 3
Landed INDEX: 5
The Hash will be :
1 1 3 4 5 9 4
```

4 1 9 3 4 5 1
Max = 5
landed = 6

Example 2 :

```
Size of hash table?
4
Give me the numbers?
3 22 17 1
MAX COLLISIONS: 2
Landed INDEX: 3
The Hash will be :
0 17 22 1 3
```

Question 2 : 10%

You are assigned to getting two sequence A and B from the users separately (We assume they are sorted).
Write the algorithm in aUnionB class which is the complexity is **O(n).**

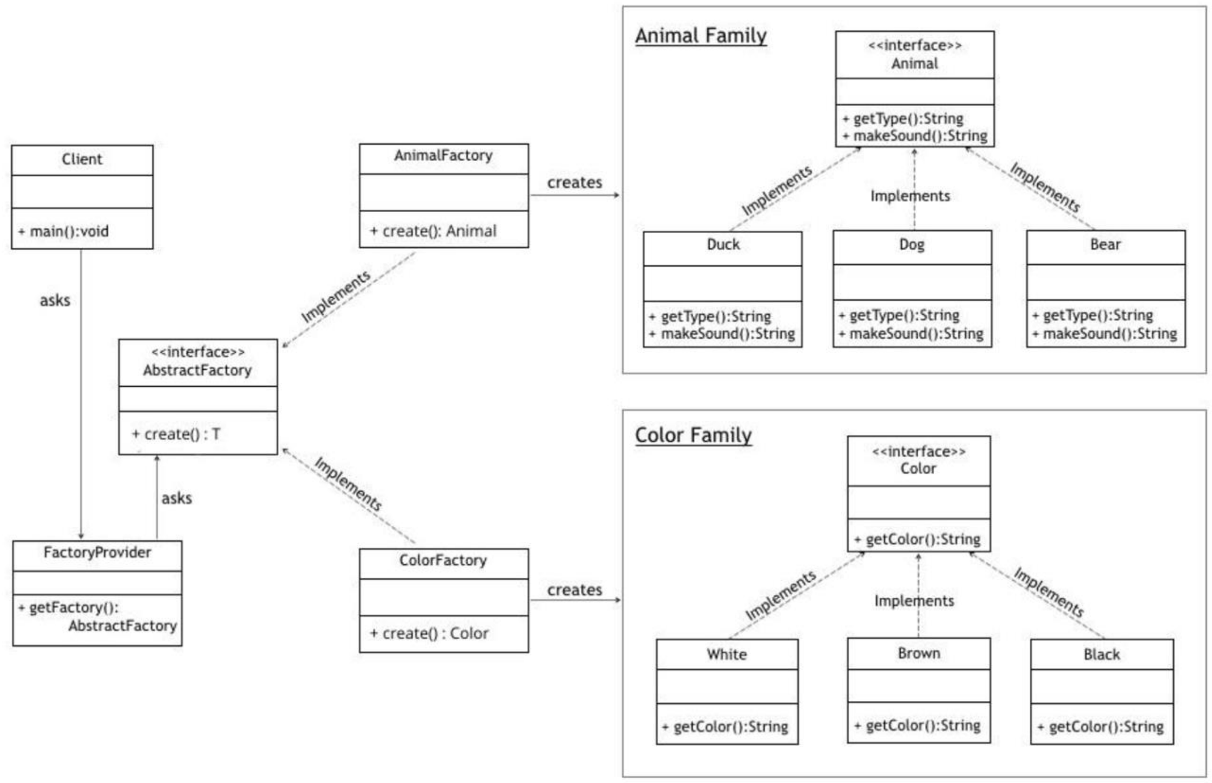
Keep in mind : for computing a sequence representing the set A U B (with no duplicates)as a sorted sequence.

```
Give me first Size?
6
Give me first seq?
1 2 2 7 9 15
Give me Second Size?
5
Give me first seq?
1 2 3 11 77

Result :

1 2 3 7 9 11 15 77
```

Question 3 : 15%

Here is the UML Diagram for abstract factory Design.

Try to implement all the class and Create method.



Question 4 : 5%

Fill the blank : For this question create the class and name Q4 and put the answers in the comments.

a.

```
@Test
public void verifyAssertion () throws InterruptedException {
WebDriver driver = new FirefoxDriver();
driver.get("https://www.irctc.co.in");
String str1 = null;
String str2 = "hello";
-------(1)-------(str1); //checking for null value
System.out.println("String holds null value – Assert passed");
}
```

b.

```
@Test

Stream<DynamicTest> testDifferentMultiplyOperations() {
        MyClass tester = new MyClass();
        int[][] data = new int[][] { { 1, 2, 2 }, { 5, 3, 15 },
        { 121, 4, 484 } };
        return Arrays.stream(data).map(entry -> {
        int m1 = entry[0];
        int m2 = entry[1];
        int expected = entry[2];
        return dynamicTest(m1 + " * " + m2 + " = " + expected, () -> {
        assertEquals(---(2)---, tester.multiply(m1, m2)); }); });
        }
```