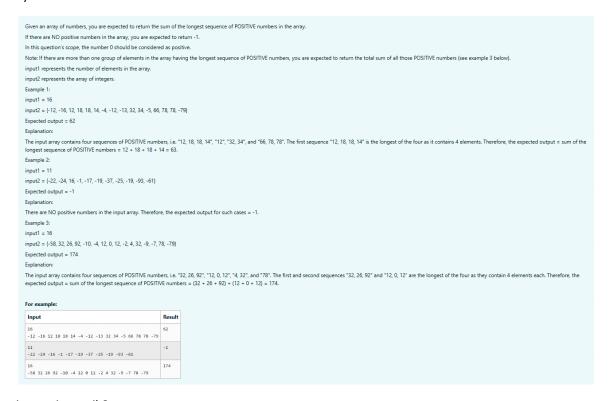
Object Oriented Programming Using Java

Week 3

1)



import java.util.Scanner;

```
public class LongestPositiveSequence {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int n = sc.nextInt();

int[] arr = new int[n];

for (int i = 0; i < n; i++) {</pre>
```

```
arr[i] = sc.nextInt();
}
int maxLen = 0, len = 0;
int maxSum = 0, sum = 0;
boolean hasPositive = false;
for (int i = 0; i < n; i++) {
  if (arr[i] >= 0) {
    hasPositive = true;
    sum += arr[i];
    len++;
  } else {
    if (len > maxLen) {
      maxLen = len;
      maxSum = sum;
    } else if (len == maxLen) {
      maxSum += sum;
    }
    sum = 0;
    len = 0;
  }
}
if (len > maxLen) {
  maxSum = sum;
} else if (len == maxLen) {
  maxSum += sum;
}
System.out.println(hasPositive ? maxSum : -1);
```

```
sc.close();
}
```

	Input	Expected	Got	
~	16 -12 -16 12 18 18 14 -4 -12 -13 32 34 -5 66 78 78 -79	62	62	~
~	11 -22 -24 -16 -1 -17 -19 -37 -25 -19 -93 -61	-1	-1	~
~	16 -58 32 26 92 -10 -4 12 0 12 -2 4 32 -9 -7 78 -79	174	174	~

2)

```
You are provided with a set of numbers (array of numbers).
You have to generate the sum of specific numbers based on its position in the array set provided to you.
This is explained below:
Example 1:
Let us assume the encoded set of numbers given to you is:
input1:5 and input2: {1, 51, 436, 7860, 41236}
Step 1:
Starting from the 0<sup>th</sup> index of the array pick up digits as per below:
0^{\text{th}} index – pick up the units value of the number (in this case is 1).
1st index - pick up the tens value of the number (in this case it is 5).
2<sup>nd</sup> index - pick up the hundreds value of the number (in this case it is 4).
3^{\rm rd} index - pick up the thousands value of the number (in this case it is 7).
4<sup>th</sup> index - pick up the ten thousands value of the number (in this case it is 4).
(Continue this for all the elements of the input array).
The array generated from Step 1 will then be – \{1, 5, 4, 7, 4\}.
Step 2:
Square each number present in the array generated in Step 1.
{1, 25, 16, 49, 16}
Step 3:
Calculate the sum of all elements of the array generated in Step 2 to get the final result. The result will be = 107.
Note:
1) While picking up a number in Step1, if you observe that the number is smaller than the required position then use 0.
2) \quad \text{ In the given function, input 1[] is the array of numbers and input 2 represents the number of elements in input 1.} \\
Example 2:
input1: 5 and input1: {1, 5, 423, 310, 61540}
Step 1:
Generating the new array based on position, we get the below array:
{1, 0, 4, 0, 6}
In this case, the value in input 1 at index 1 and 3 is less than the value required to be picked up based on position, so we use a 0.
Step 2:
{1, 0, 16, 0, 36}
Step 3:
The final result = 53.
For example:
                         Result
 Input
 1 51 436 7860 41236
 1 5 423 310 61540
```

import java.util.Scanner;

```
public class ArraySum {
```

```
public static void main(String[] args) {
     Scanner sc = new Scanner(System.in);
     int n = sc.nextInt();
     int[] input1 = new int[n];
     for (int i = 0; i < n; i++) {
       input1[i] = sc.nextInt();
     }
    int[] newArray = new int[n];
    for (int i = 0; i < n; i++) {
       int number = input1[i];
       int digit = 0;
       if (i == 0) {
         digit = number % 10;
       } else if (i == 1) {
         digit = (number / 10) % 10;
       } else if (i == 2) {
         digit = (number / 100) % 10;
       } else if (i == 3) {
         digit = (number / 1000) % 10;
       } else if (i == 4) {
         digit = (number / 10000) % 10;
       }
       if (number < Math.pow(10, i)) {</pre>
         digit = 0;
       }
       newArray[i] = digit;
     }
    for (int i = 0; i < n; i++) {
```

```
newArray[i] = newArray[i] * newArray[i];
}
int sum = 0;
for (int i = 0; i < n; i++) {
    sum += newArray[i];
}
System.out.println(sum);
sc.close();
}</pre>
```