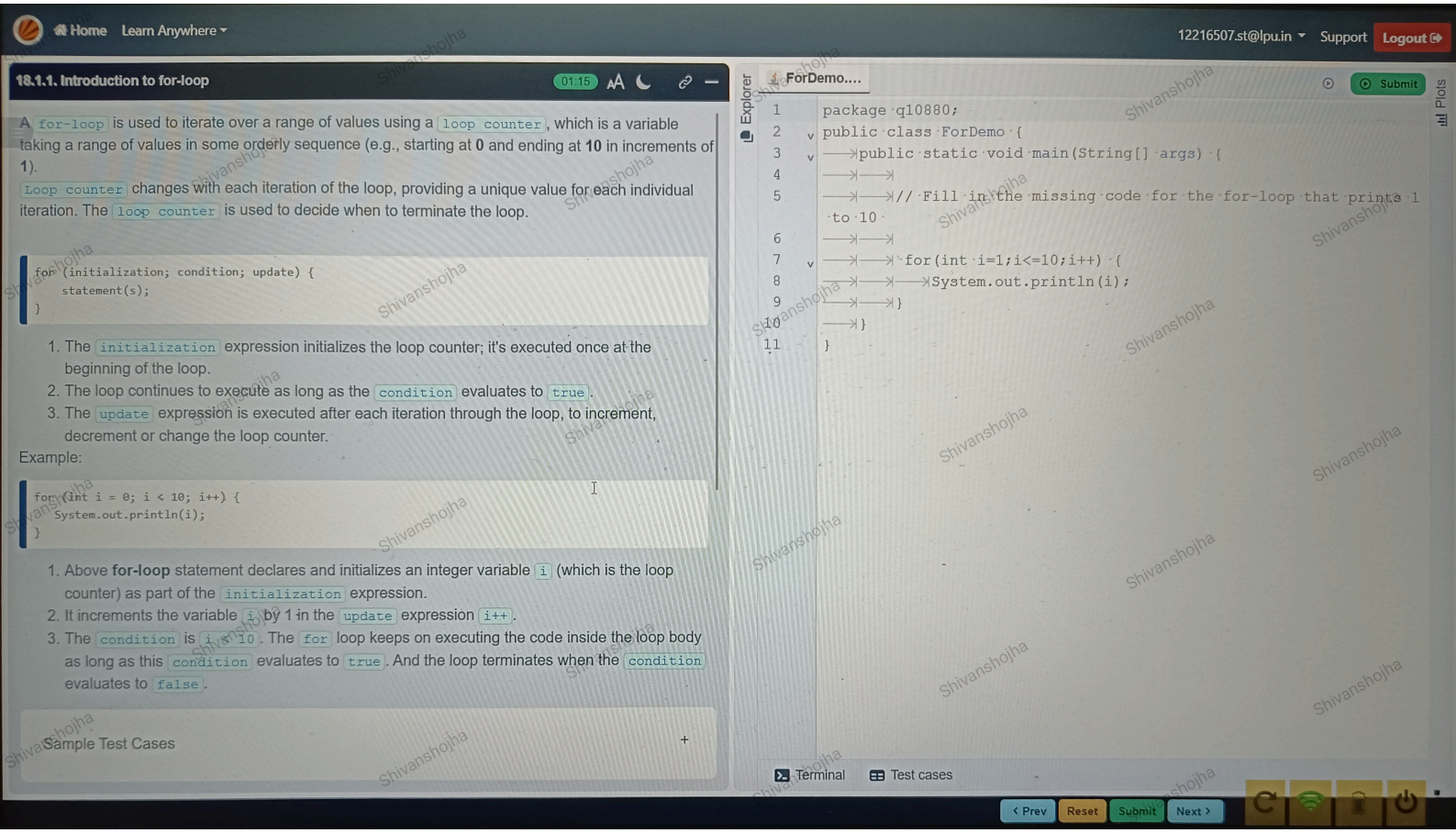
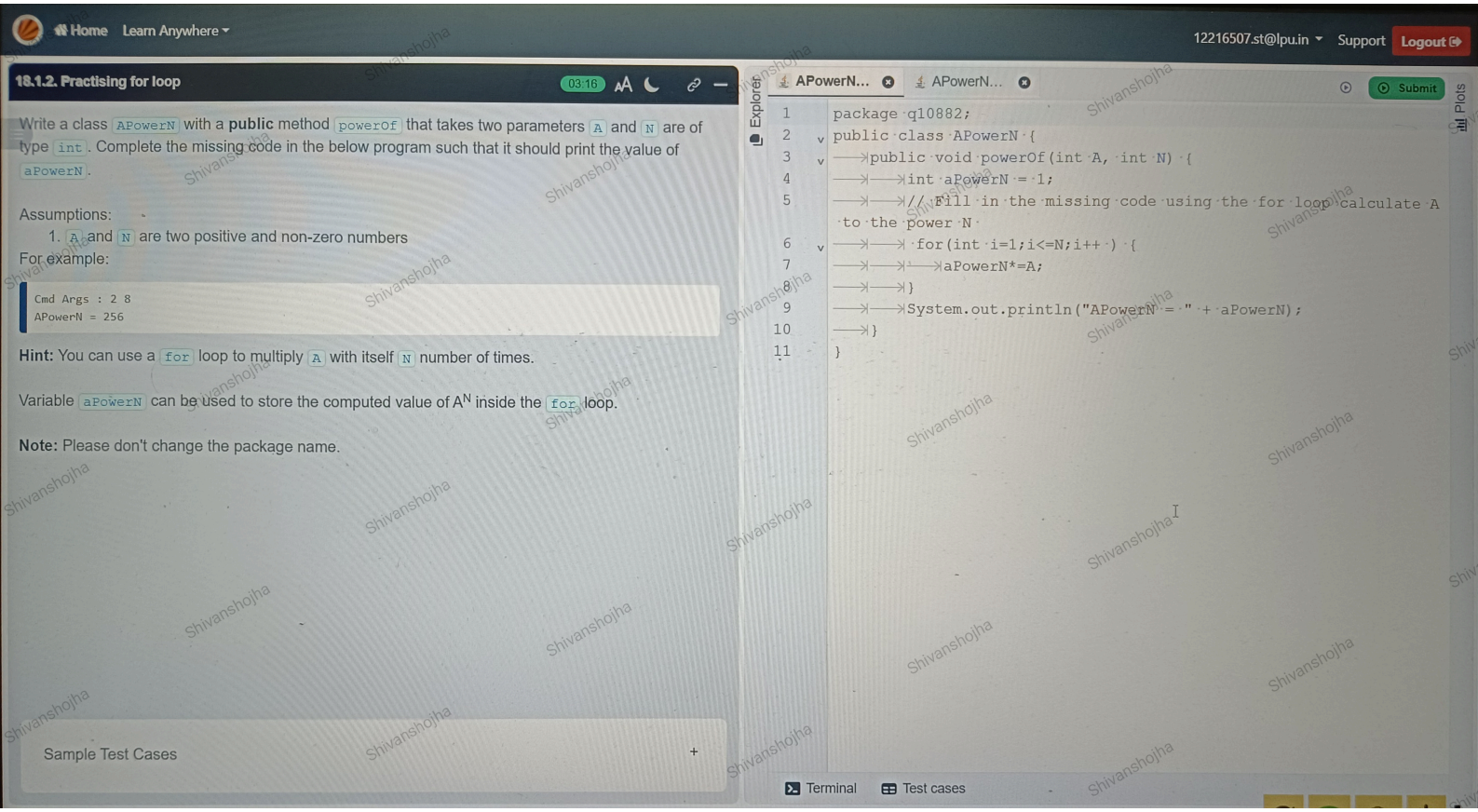
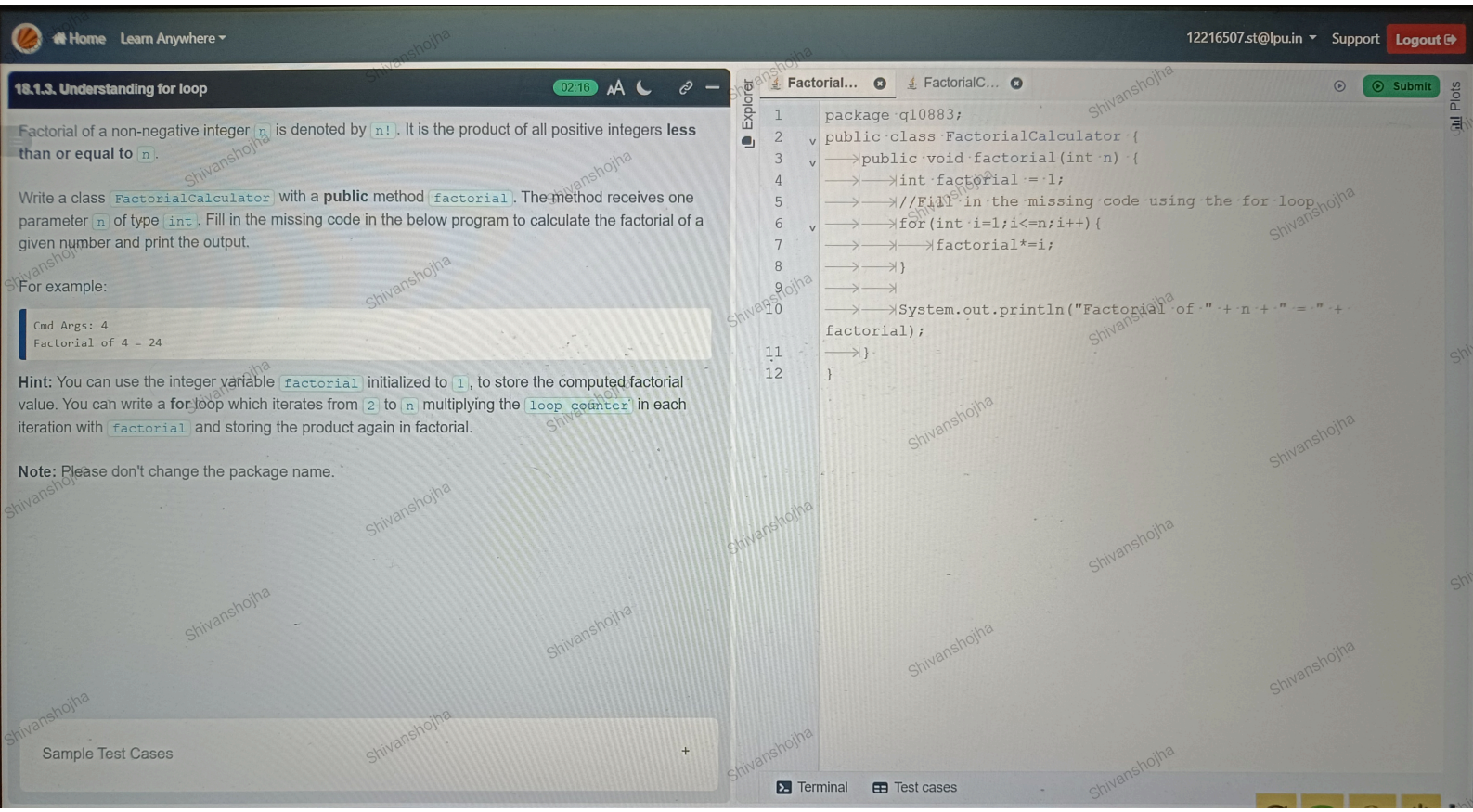
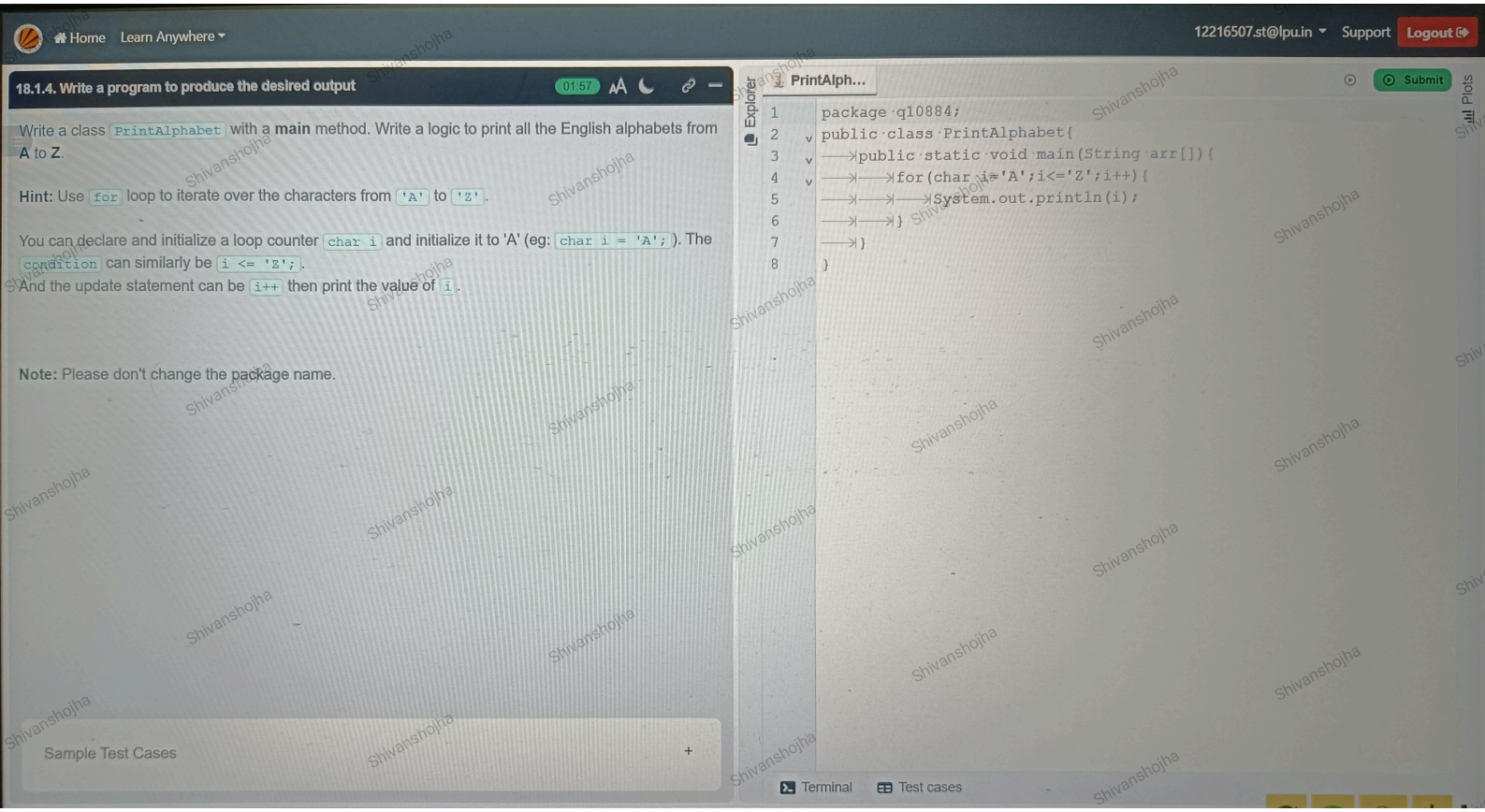
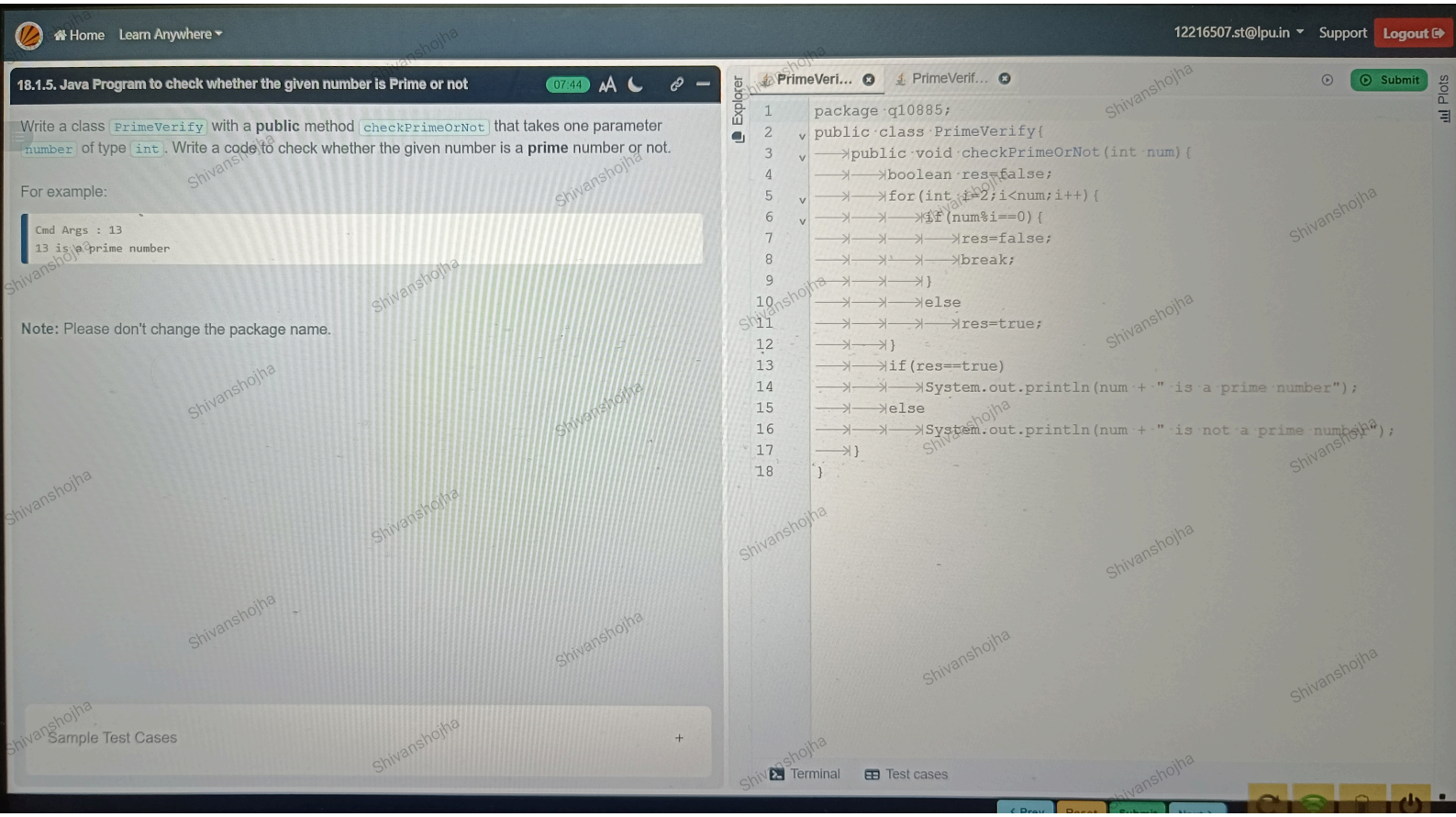
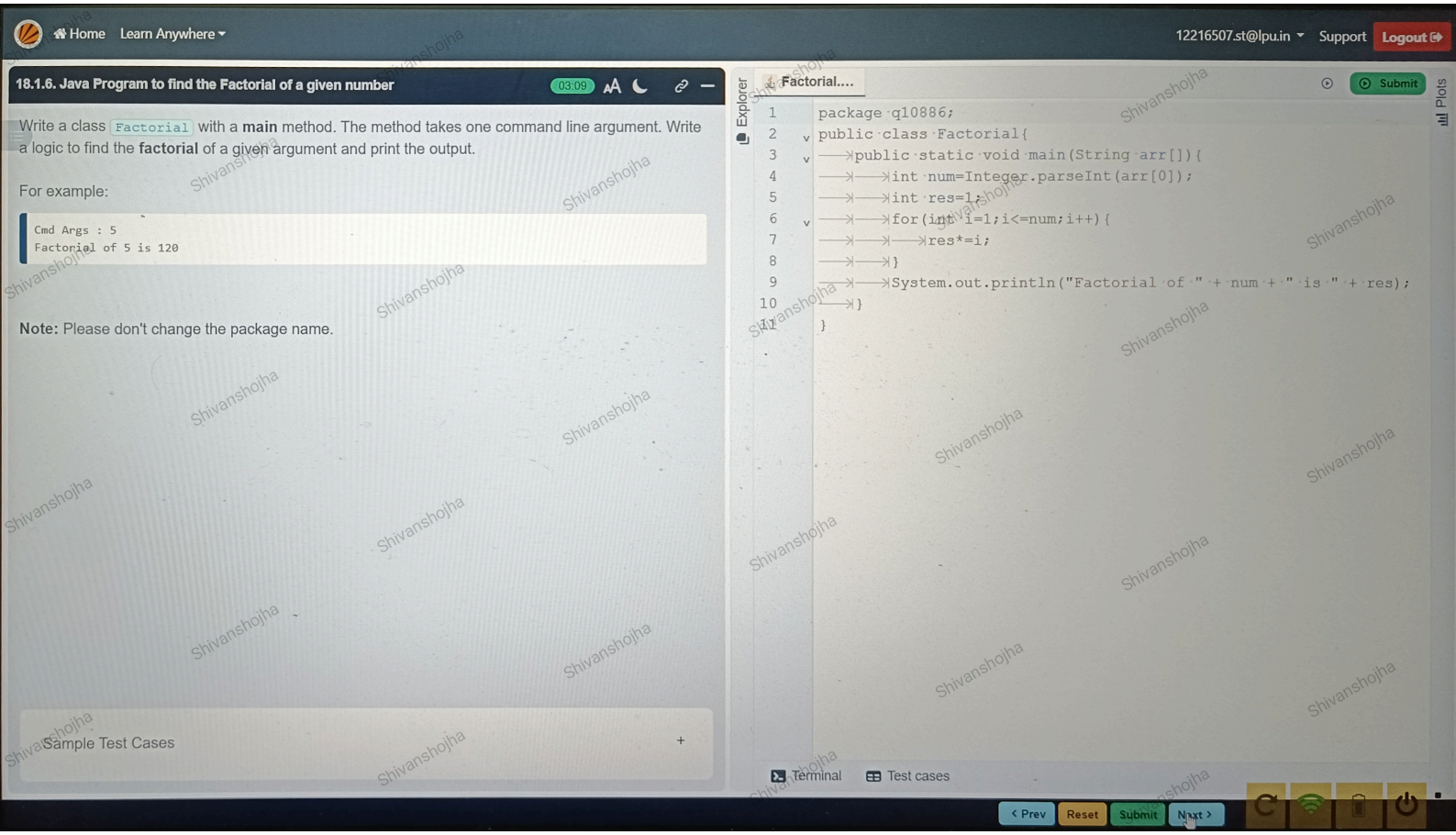

Home
Learn Anywhere
12216507.st@lpu.in
Support
Logout
18.1.1. Introduction to for-loop
01:15
A for-loop is used to iterate over a range of values using a loop counter, which is a variable taking a range of values in some orderly sequence (e.g., starting at 0 and ending at 10 in increments of 1).
Loop counter changes with each iteration of the loop, providing a unique value for each individual iteration. The loop counter is used to decide when to terminate the loop.
for (initialization; condition; update) {
statement(s);
}
1. The initialization expression initializes the loop counter; it's executed once at the beginning of the loop.
2. The loop continues to execute as long as the condition evaluates to true.
3. The update expression is executed after each iteration through the loop, to increment, decrement or change the loop counter.
Example:
for (int i = 0; i < 10; i++) {
System.out.println(i);
}
1. Above for-loop statement declares and initializes an integer variable i (which is the loop counter) as part of the initialization expression.
2. It increments the variable i by 1 in the update expression i++.
3. The condition is i < 10. The for loop keeps on executing the code inside the loop body as long as this condition evaluates to true. And the loop terminates when the condition evaluates to false.
Sample Test Cases
Explorer
ForDemo....
Submit
Plots
1 package q10880;
2 public class ForDemo {
3 public static void main(String[] args) {
4
5 // Fill in the missing code for the for-loop that prints 1 to 10
6
7 for(int i=1;i<=10;i++) {
8 System.out.println(i);
9 }
10 }
11 }
Terminal
Test cases
Prev
Reset
Submit
Next

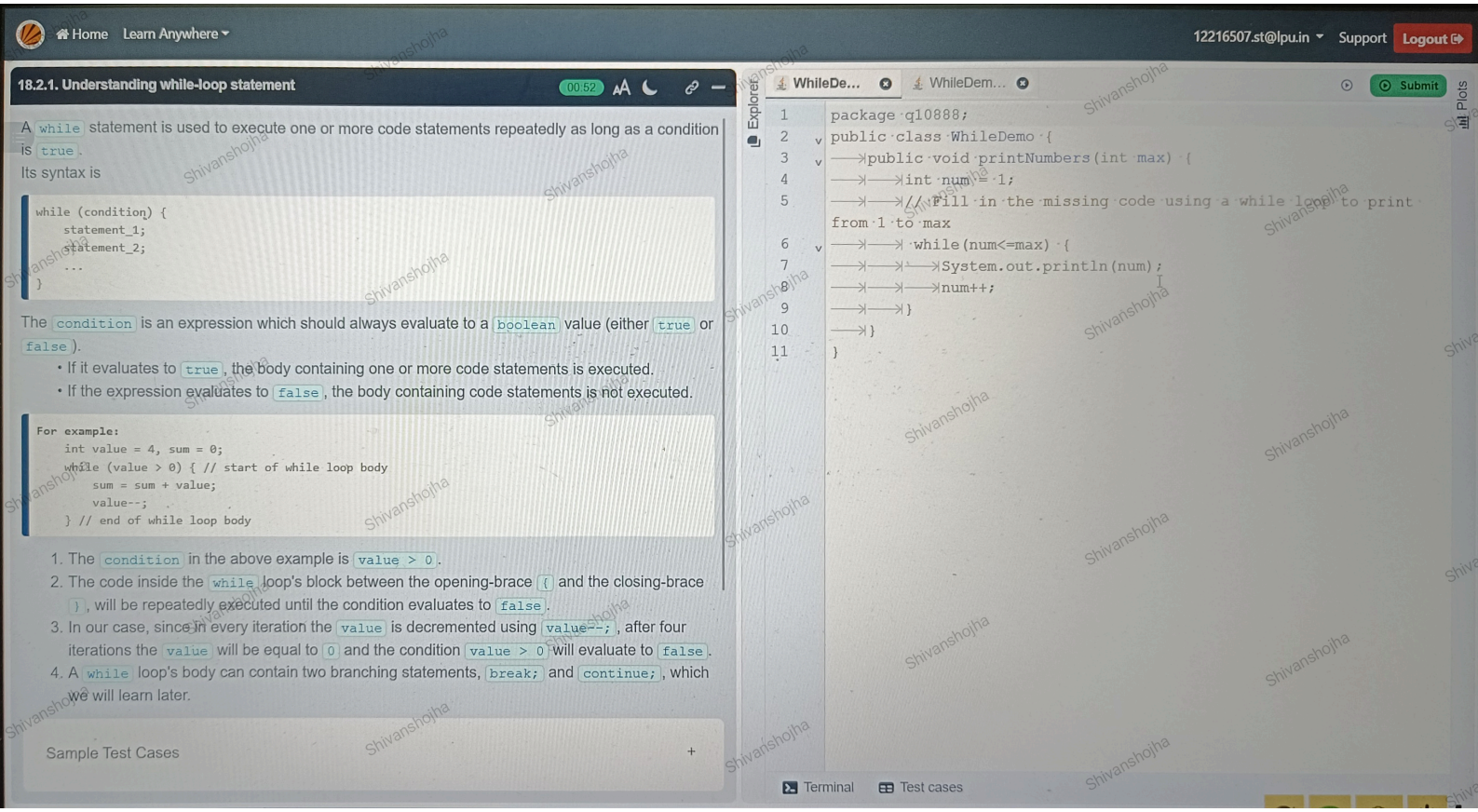

Home
Learn Anywhere
12216507.st@lpu.in
Support
Logout
18.1.2. Practising for loop
03:16
Write a class APowerN with a public method powerOf that takes two parameters A and N are of type int. Complete the missing code in the below program such that it should print the value of aPowerN.
Assumptions:
1. A and N are two positive and non-zero numbers
For example:
Cmd Args : 2 8
APowerN = 256
Hint: You can use a for loop to multiply A with itself N number of times.
Variable aPowerN can be used to store the computed value of A^N inside the for loop.
Note: Please don't change the package name.
Sample Test Cases
Explorer
APowerN...
APowerN...
Submit
Plots
1 package q10882;
2 public class APowerN {
3 public void powerOf(int A, int N) {
4 int aPowerN = 1;
5 // Fill in the missing code using the for loop calculate A to the power N
6 for(int i=1;i<=N;i++) {
7 aPowerN*=A;
8 }
9 System.out.println("APowerN = " + aPowerN);
10 }
11 }
Terminal
Test cases

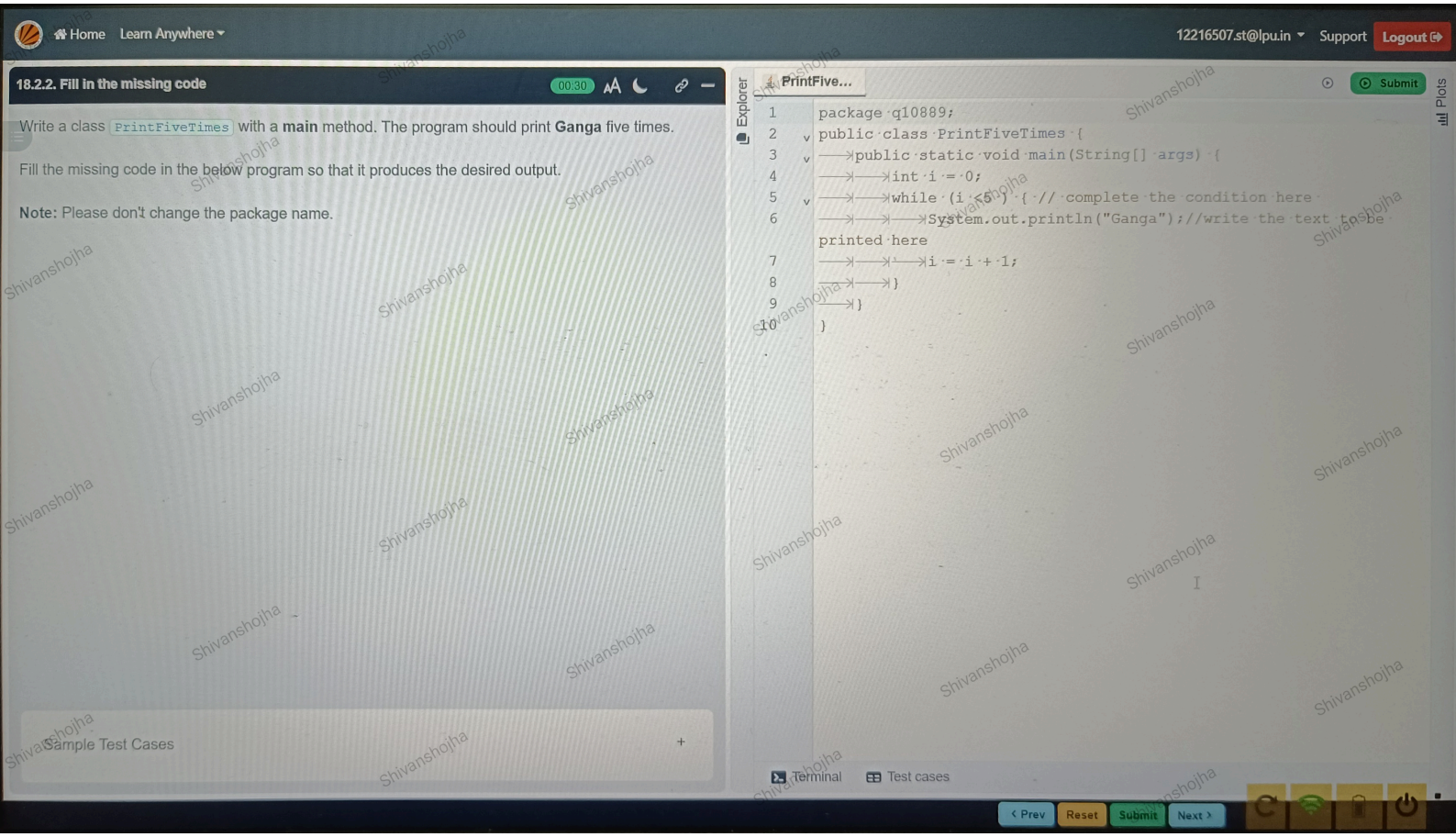

Home
Learn Anywhere
12216507.st@lpu.in
Support
Logout
18.1.3. Understanding for loop
02:16
Factorial of a non-negative integer n is denoted by n!. It is the product of all positive integers less than or equal to n.
Write a class FactorialCalculator with a public method factorial. The method receives one parameter n of type int. Fill in the missing code in the below program to calculate the factorial of a given number and print the output.
For example:
Cmd Args: 4
Factorial of 4 = 24
Hint: You can use the integer variable factorial initialized to 1, to store the computed factorial value. You can write a for loop which iterates from 2 to n multiplying the loop counter in each iteration with factorial and storing the product again in factorial.
Note: Please don't change the package name.
Sample Test Cases
Explorer
Factorial...
FactorialC...
Submit
Plots
1 package q10883;
2 public class FactorialCalculator {
3 public void factorial(int n) {
4 int factorial = 1;
5 //Fill in the missing code using the for loop
6 for(int i=1;i<=n;i++){
7 factorial*=i;
8 }
9
10 System.out.println("Factorial of " + n + " = " + factorial);
11 }
12 }
Terminal
Test cases

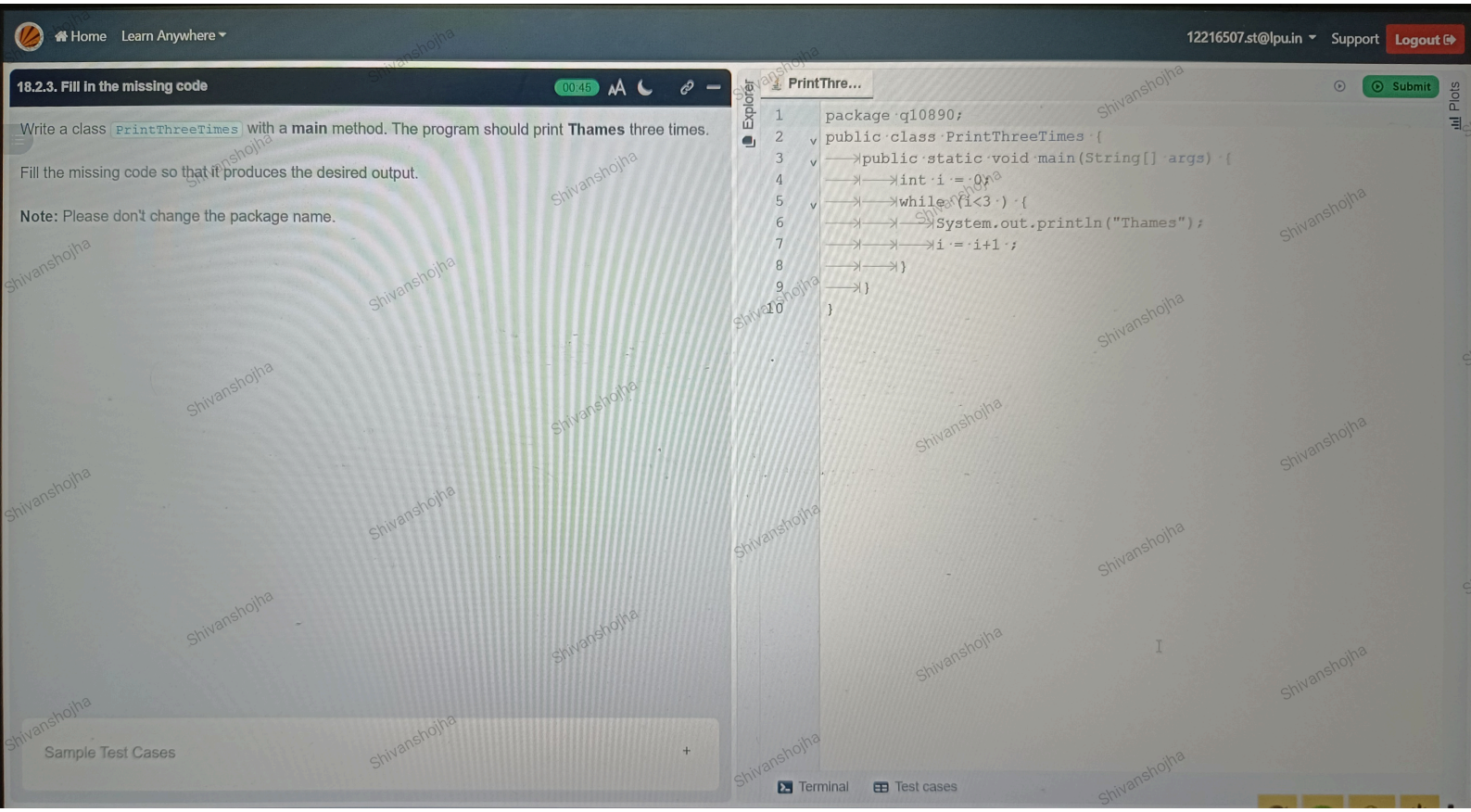

Home
Learn Anywhere
12216507.st@lpu.in
Support
Logout
18.1.4. Write a program to produce the desired output
01:57
Write a class PrintAlphabet with a main method. Write a logic to print all the English alphabets from A to Z.
Hint: Use for loop to iterate over the characters from 'A' to 'Z'.
You can declare and initialize a loop counter char i and initialize it to 'A' (eg: char i = 'A';). The condition can similarly be i <= 'Z';.
And the update statement can be i++ then print the value of i.
Note: Please don't change the package name.
Sample Test Cases
Explorer
PrintAlph...
Submit
Plots
1 package q10884;
2 public class PrintAlphabet{
3 public static void main(String arr[]){
4 for(char i='A';i<='Z';i++){
5 System.out.println(i);
6 }
7 }
8 }
Terminal
Test cases

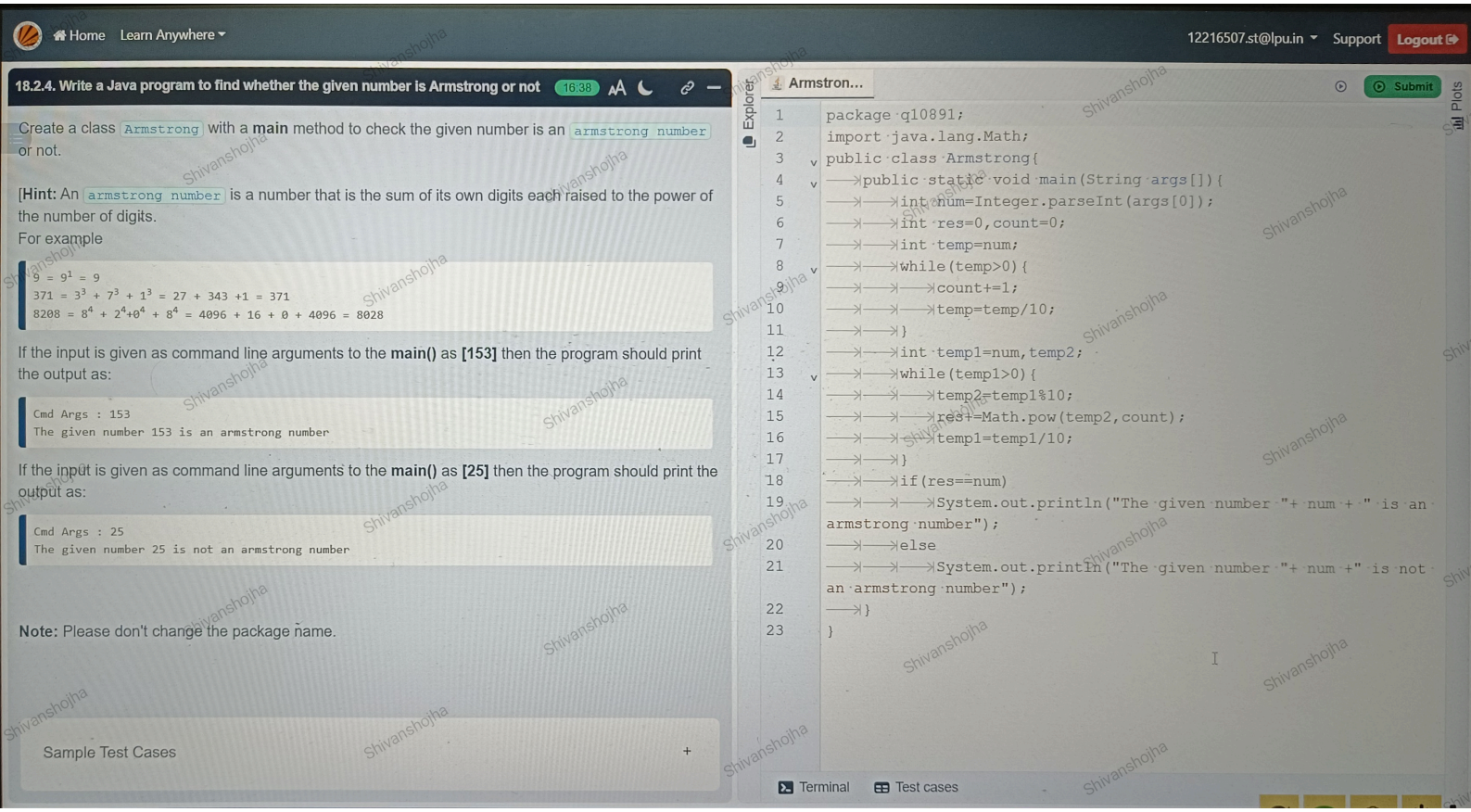

Home
Learn Anywhere
12216507.st@lpu.in
Support
Logout
18.1.5. Java Program to check whether the given number is Prime or not
07:44
Write a class PrimeVerify with a public method checkPrimeOrNot that takes one parameter number of type int. Write a code to check whether the given number is a prime number or not.
For example:
Cmd Args : 13
13 is a prime number
Note: Please don't change the package name.
Sample Test Cases
Explorer
PrimeVeri...
PrimeVerif...
Submit
Plots
1 package q10885;
2 public class PrimeVerify{
3 public void checkPrimeOrNot(int num){
4 boolean res=false;
5 for(int i=2;i<num;i++){
6 if(num%i==0){
7 res=false;
8 break;
9 }
10 else
11 res=true;
12 }
13 if(res==true)
14 System.out.println(num + " is a prime number");
15 else
16 System.out.println(num + " is not a prime number");
17 }
18 }
Terminal
Test cases

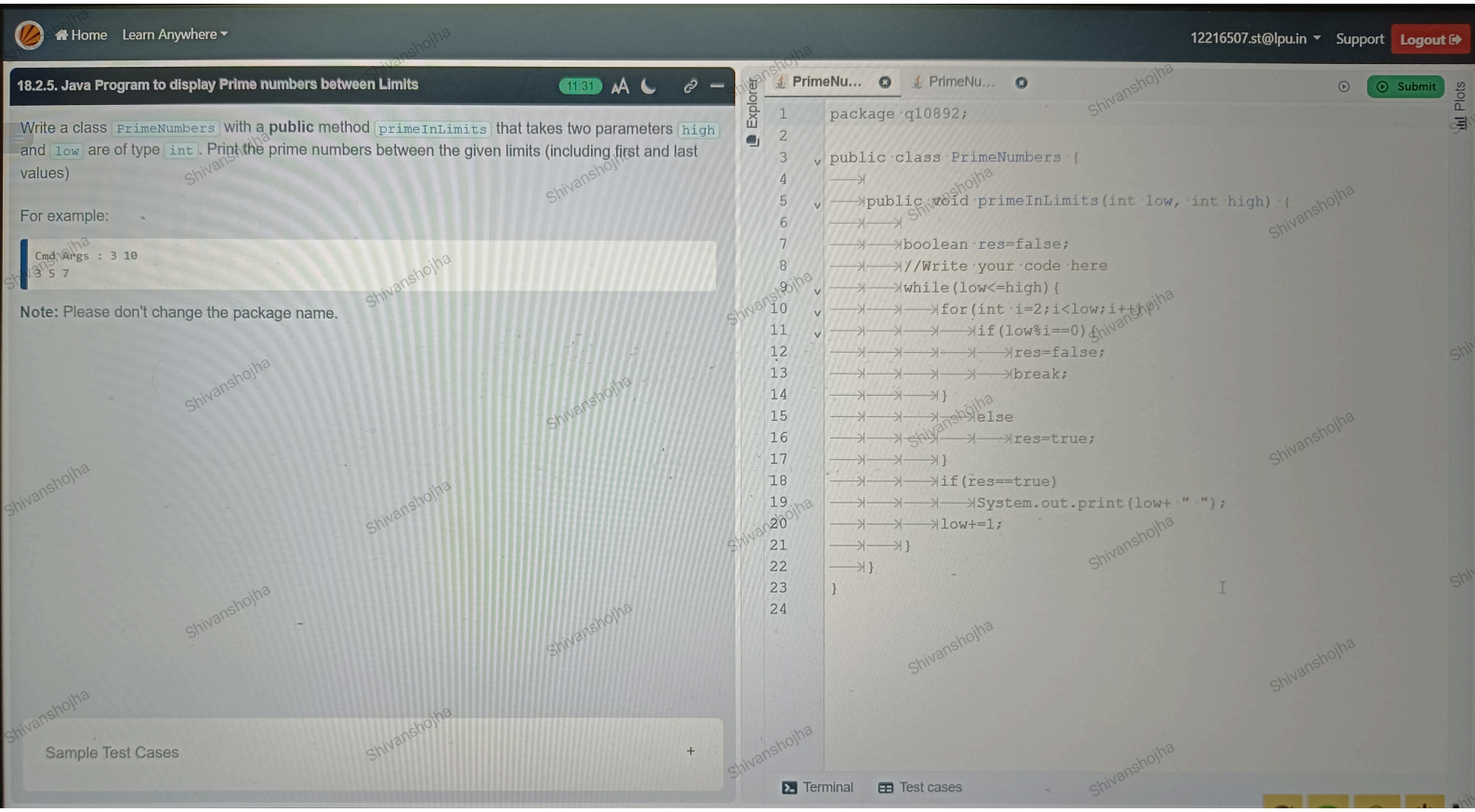

Home
Learn Anywhere
12216507.st@lpu.in
Support
Logout
18.1.6. Java Program to find the Factorial of a given number
03:09
Write a class Factorial with a main method. The method takes one command line argument. Write a logic to find the factorial of a given argument and print the output.
For example:
Cmd Args : 5
Factorial of 5 is 120
Note: Please don't change the package name.
Sample Test Cases
Explorer
Factorial....
Submit
Plots
1 package q10886;
2 public class Factorial{
3 public static void main(String arr[]){
4 int num=Integer.parseInt(arr[0]);
5 int res=1;
6 for(int i=1;i<=num;i++){
7 res*=i;
8 }
9 System.out.println("Factorial of " + num + " is " + res);
10 }
11 }
Terminal
Test cases
< Prev
Reset
Submit
Next >

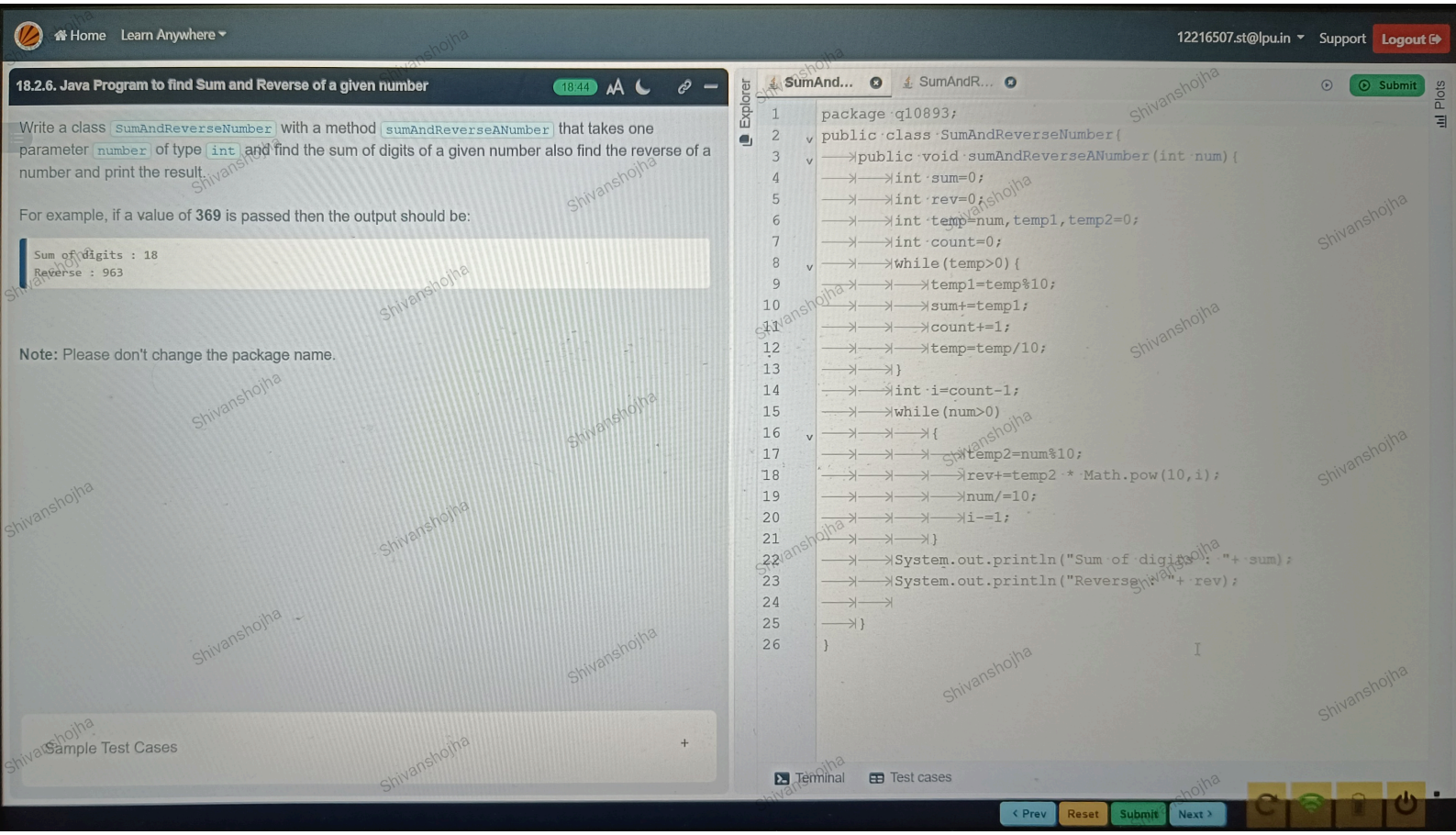

Home
Learn Anywhere
12216507.st@lpu.in
Support
Logout
18.2.1. Understanding while-loop statement
00:52
A while statement is used to execute one or more code statements repeatedly as long as a condition is true.
Its syntax is
while (condition) {
statement_1;
statement_2;
...
}
The condition is an expression which should always evaluate to a boolean value (either true or false).
• If it evaluates to true, the body containing one or more code statements is executed.
• If the expression evaluates to false, the body containing code statements is not executed.
For example:
int value = 4, sum = 0;
while (value > 0) { // start of while loop body
sum = sum + value;
value--;
} // end of while loop body
1. The condition in the above example is value > 0.
2. The code inside the while loop's block between the opening-brace { and the closing-brace }, will be repeatedly executed until the condition evaluates to false.
3. In our case, since in every iteration the value is decremented using value--;, after four iterations the value will be equal to 0 and the condition value > 0 will evaluate to false.
4. A while loop's body can contain two branching statements, break; and continue;, which we will learn later.
Sample Test Cases
Explorer
WhileDe...
WhileDem...
Submit
Plots
1 package q10888;
2 public class WhileDemo {
3 public void printNumbers(int max) {
4 int num = 1;
5 //Fill in the missing code using a while loop to print from 1 to max
6 while(num<=max) {
7 System.out.println(num);
8 num++;
9 }
10 }
11 }
Terminal
Test cases

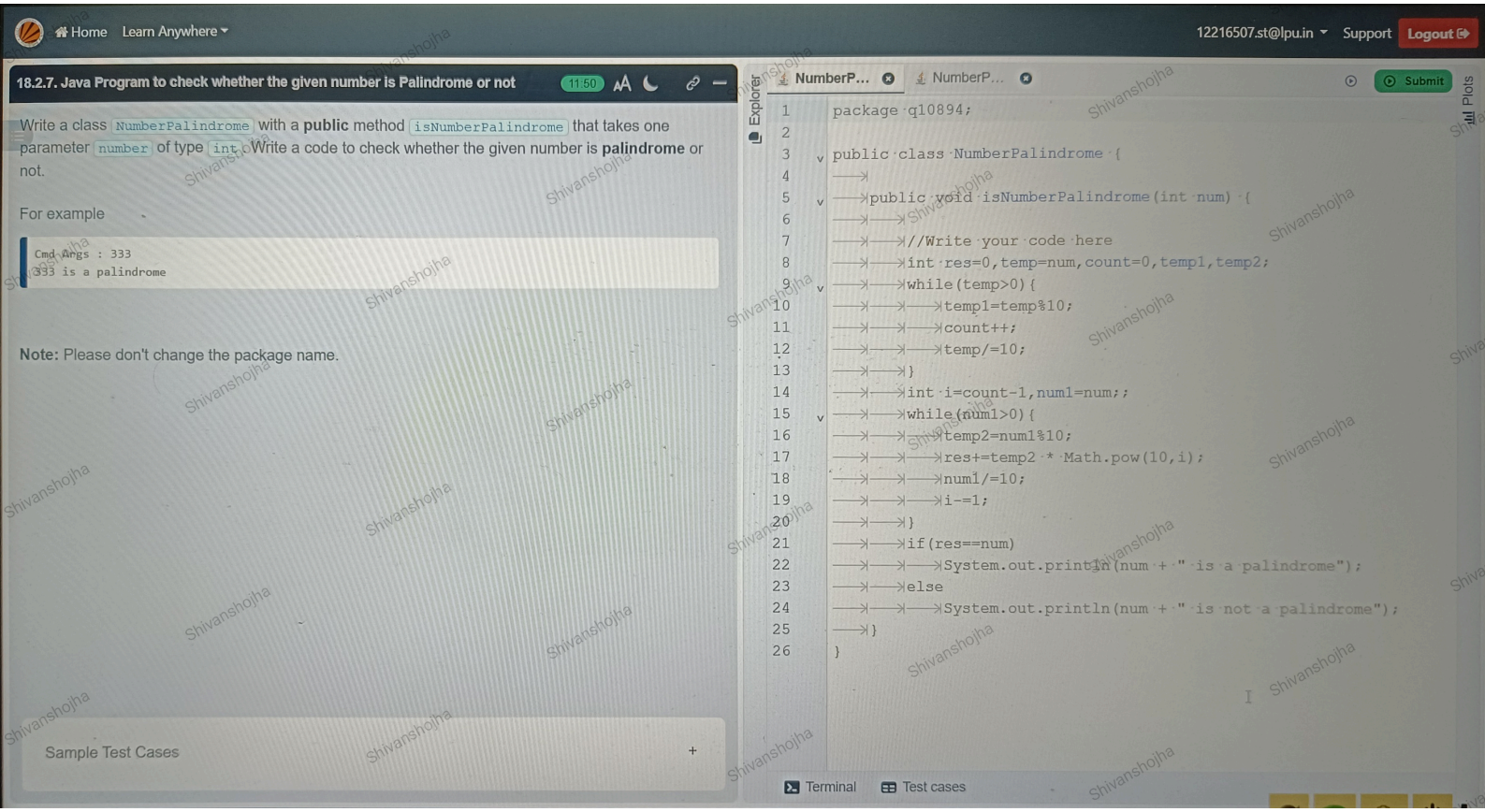

Home
Learn Anywhere
12216507.st@lpu.in
Support
Logout
18.2.2. Fill in the missing code
00:30
Write a class PrintFiveTimes with a main method. The program should print Ganga five times.
Fill the missing code in the below program so that it produces the desired output.
Note: Please don't change the package name.
Sample Test Cases
PrintFive...
Submit
Explorer
Plots
1 package q10889;
2 public class PrintFiveTimes {
3 public static void main(String[] args) {
4 int i = 0;
5 while (i <5) { // complete the condition here
6 System.out.println("Ganga");//write the text to be printed here
7 i = i + 1;
8 }
9 }
10 }
Terminal
Test cases
Prev
Reset
Submit
Next

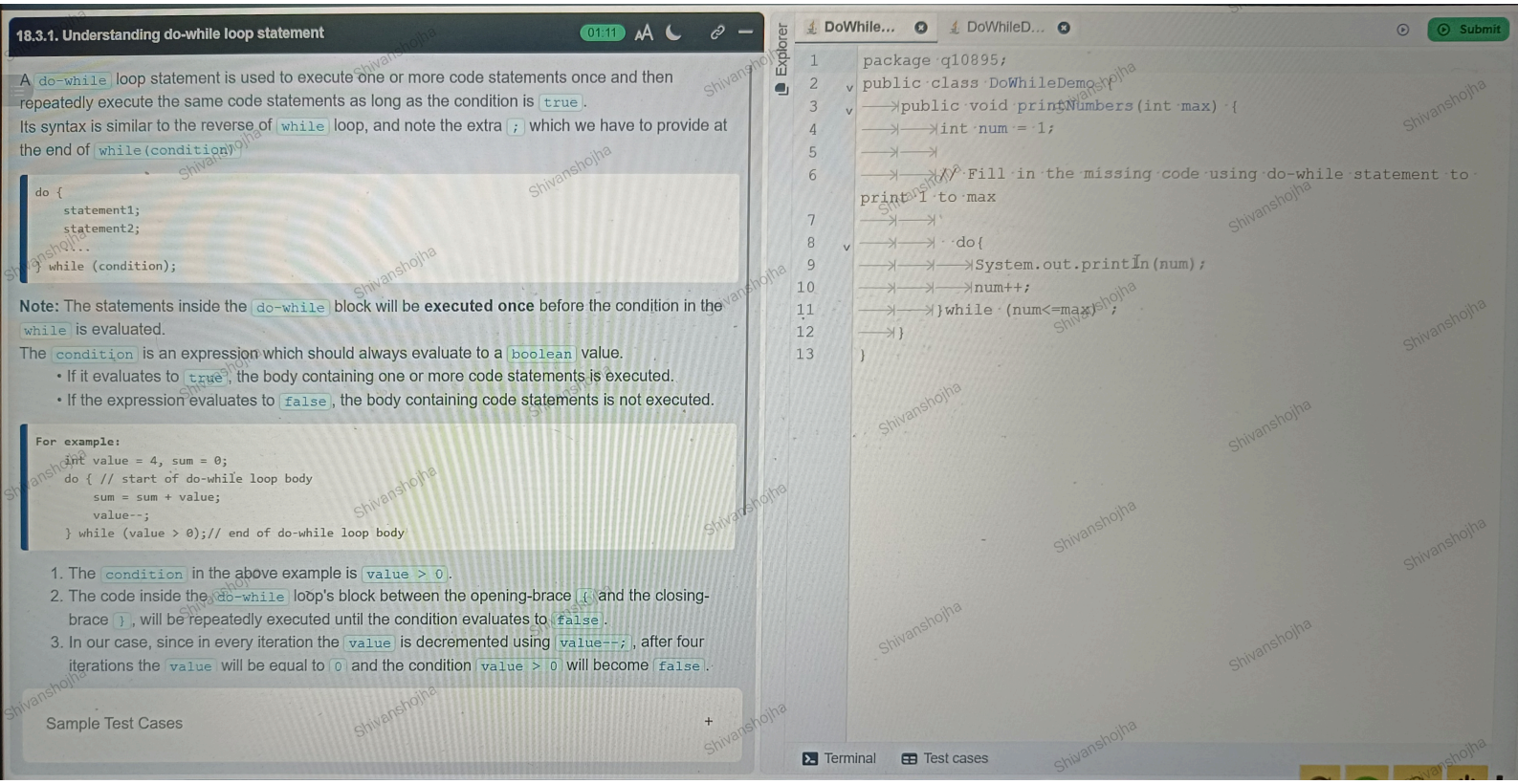

Home
Learn Anywhere
12216507.st@lpu.in
Support
Logout
18.2.3. Fill in the missing code
00:45
Write a class PrintThreeTimes with a main method. The program should print Thames three times.
Fill the missing code so that it produces the desired output.
Note: Please don't change the package name.
Sample Test Cases
Explorer
PrintThre...
Submit
Plots
1 package q10890;
2 public class PrintThreeTimes {
3 public static void main(String[] args) {
4 int i = 0;
5 while (i<3) {
6 System.out.println("Thames");
7 i = i+1 ;
8 }
9 }
10 }
Terminal
Test cases

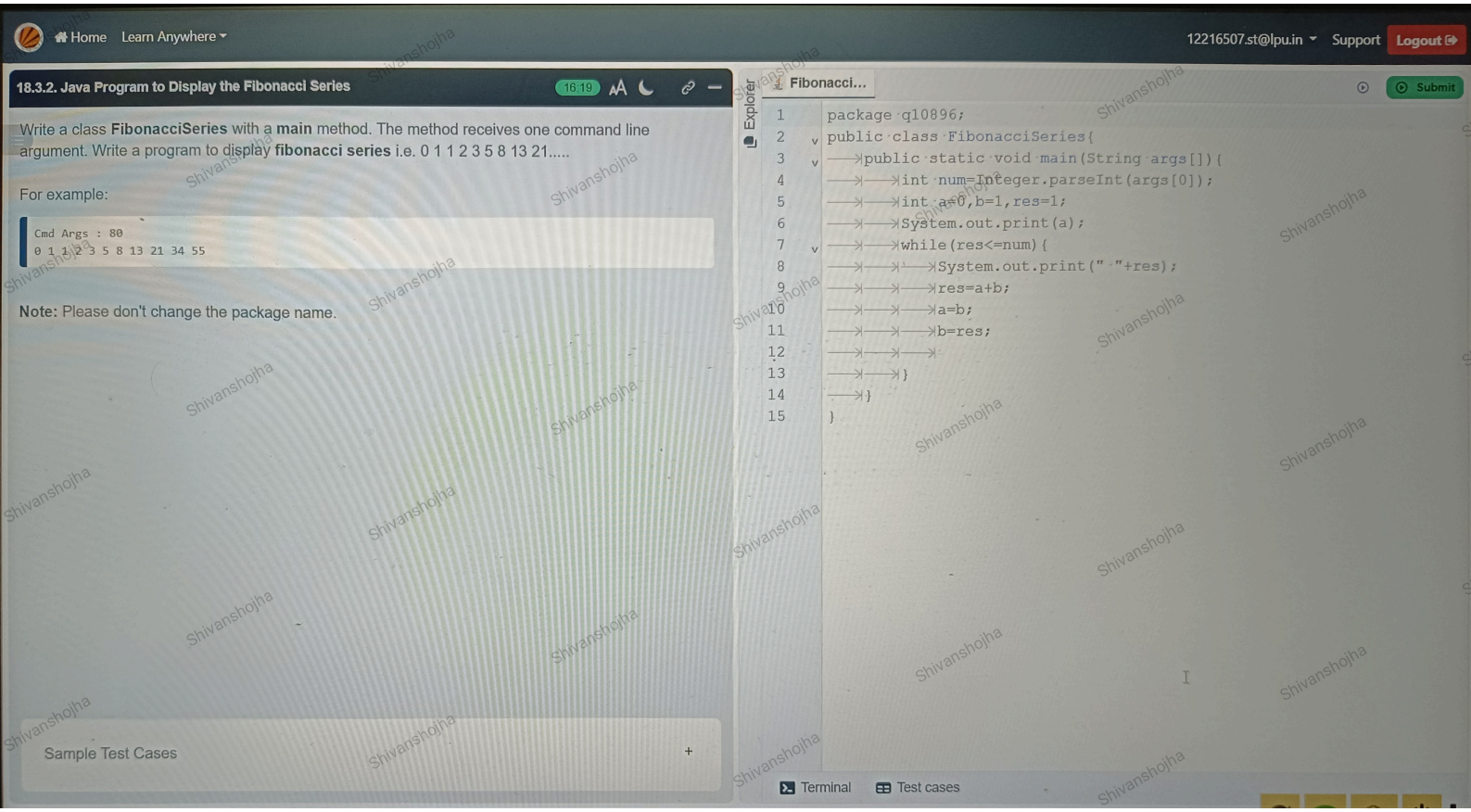

Home
Learn Anywhere
12216507.st@lpu.in
Support
Logout
18.2.4. Write a Java program to find whether the given number is Armstrong or not
16:38
Create a class Armstrong with a main method to check the given number is an armstrong number or not.
[Hint: An armstrong number is a number that is the sum of its own digits each raised to the power of the number of digits.
For example
9 = 9^1 = 9
371 = 3^3 + 7^3 + 1^3 = 27 + 343 +1 = 371
8208 = 8^4 + 2^4+0^4 + 8^4 = 4096 + 16 + 0 + 4096 = 8028
If the input is given as command line arguments to the main() as [153] then the program should print the output as:
Cmd Args : 153
The given number 153 is an armstrong number
If the input is given as command line arguments to the main() as [25] then the program should print the output as:
Cmd Args : 25
The given number 25 is not an armstrong number
Note: Please don't change the package name.
Sample Test Cases
Explorer
Armstron...
Submit
Plots
1 package q10891;
2 import java.lang.Math;
3 public class Armstrong{
4 public static void main(String args[]){
5 int num=Integer.parseInt(args[0]);
6 int res=0,count=0;
7 int temp=num;
8 while(temp>0){
9 count+=1;
10 temp=temp/10;
11 }
12 int temp1=num,temp2;
13 while(temp1>0){
14 temp2=temp1%10;
15 res+=Math.pow(temp2,count);
16 temp1=temp1/10;
17 }
18 if(res==num)
19 System.out.println("The given number "+ num + " is an armstrong number");
20 else
21 System.out.println("The given number "+ num +" is not an armstrong number");
22 }
23 }
Terminal
Test cases

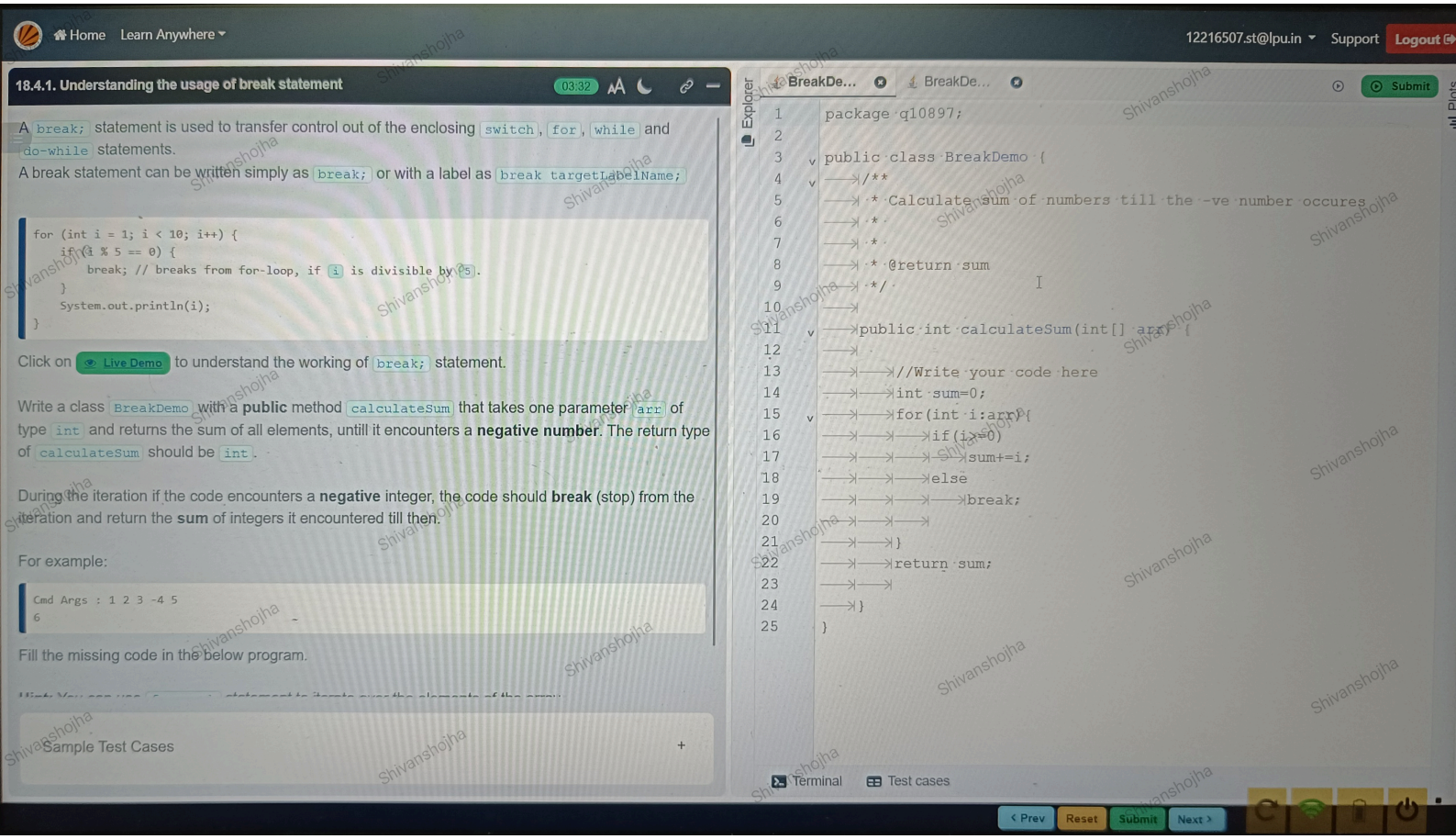

Home
Learn Anywhere
12216507.st@lpu.in
Support
Logout
18.2.5. Java Program to display Prime numbers between Limits
11:31
Write a class PrimeNumbers with a public method primeInLimits that takes two parameters high and low are of type int. Print the prime numbers between the given limits (including first and last values)
For example:
Cmd Args : 3 10
3 5 7
Note: Please don't change the package name.
Sample Test Cases
Explorer
PrimeNu...
PrimeNu...
Submit
Plots
1 package q10892;
2
3 public class PrimeNumbers {
4
5 public void primeInLimits(int low, int high) {
6
7 boolean res=false;
8 //Write your code here
9 while(low<=high){
10 for(int i=2;i<low;i++){
11 if(low%i==0){
12 res=false;
13 break;
14 }
15 else
16 res=true;
17 }
18 if(res==true)
19 System.out.print(low+" ");
20 low+=1;
21 }
22 }
23 }
24
Terminal
Test cases

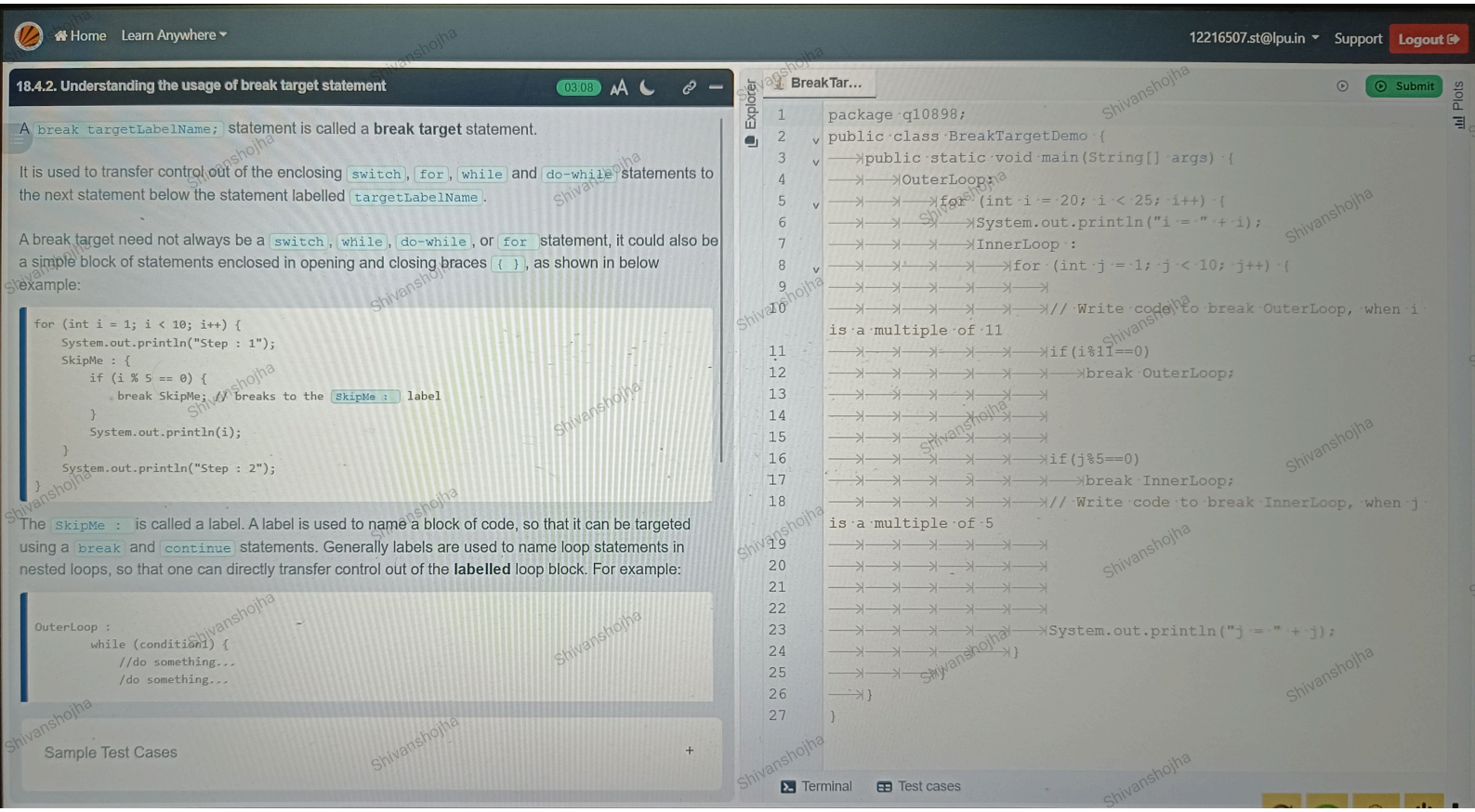

Home
Learn Anywhere
12216507.st@lpu.in
Support
Logout
18.2.6. Java Program to find Sum and Reverse of a given number
18:44
Write a class SumAndReverseNumber with a method sumAndReverseANumber that takes one parameter number of type int and find the sum of digits of a given number also find the reverse of a number and print the result.
For example, if a value of 369 is passed then the output should be:
Sum of digits : 18
Reverse : 963
Note: Please don't change the package name.
Sample Test Cases
SumAnd...
SumAndR...
Submit
Explorer
Plots
package q10893;
public class SumAndReverseNumber{
public void sumAndReverseANumber(int num){
int sum=0;
int rev=0;
int temp=num,temp1,temp2=0;
int count=0;
while(temp>0){
temp1=temp%10;
sum+=temp1;
count+=1;
temp=temp/10;
}
int i=count-1;
while(num>0)
{
temp2=num%10;
rev+=temp2 * Math.pow(10,i);
num/=10;
i-=1;
}
System.out.println("Sum of digits : "+ sum);
System.out.println("Reverse :" + rev);
}
}
Terminal
Test cases
Prev
Reset
Submit
Next

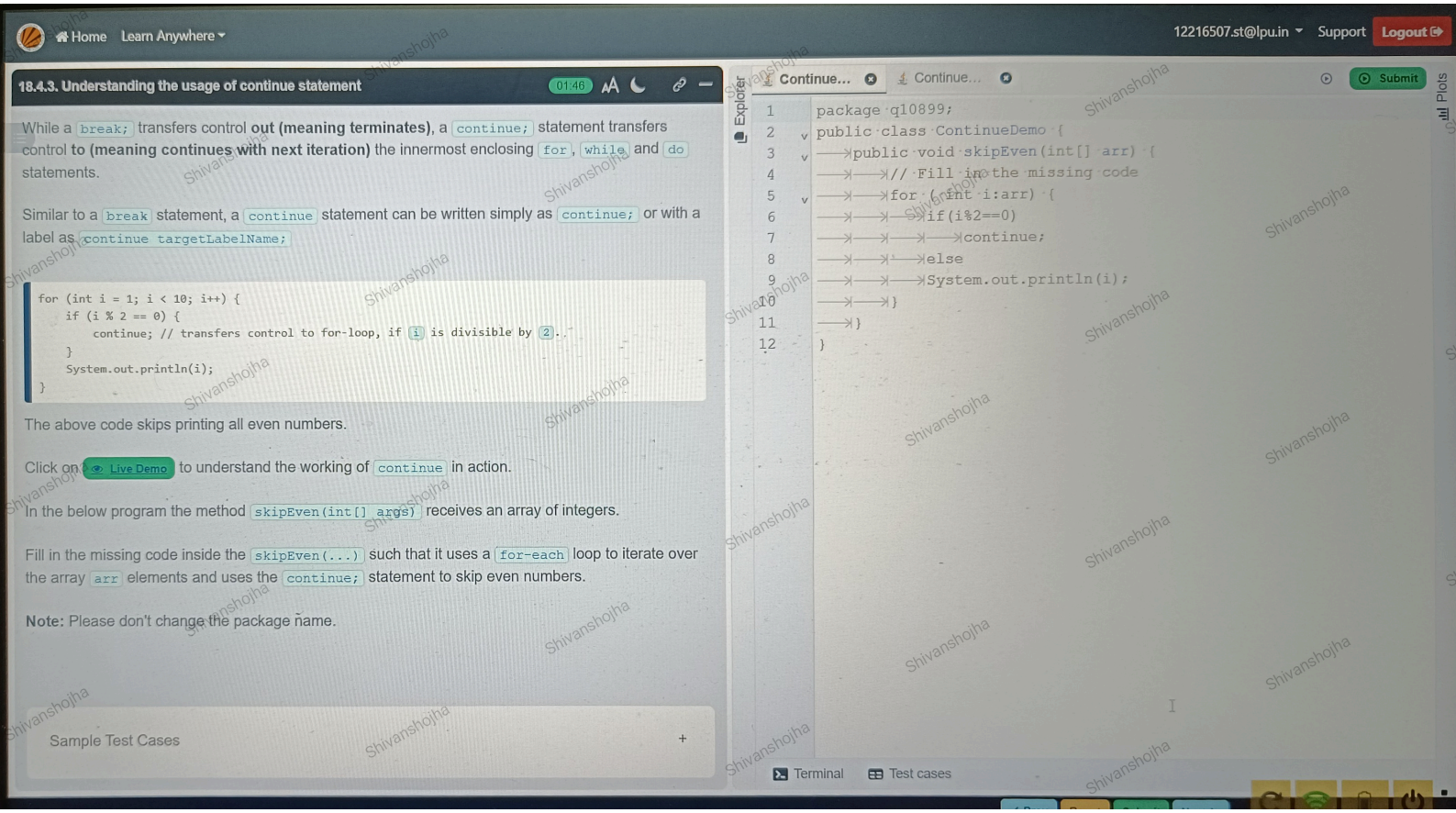

Home
Learn Anywhere
12216507.st@lpu.in
Support
Logout
18.2.7. Java Program to check whether the given number is Palindrome or not
11:50
Write a class NumberPalindrome with a public method isNumberPalindrome that takes one parameter number of type int. Write a code to check whether the given number is palindrome or not.
For example
Cmd Args : 333
333 is a palindrome
Note: Please don't change the package name.
Sample Test Cases
NumberP...
NumberP...
Submit
Explorer
Plots
1 package q10894;
2
3 public class NumberPalindrome {
4
5 public void isNumberPalindrome(int num) {
6
7 //Write your code here
8 int res=0,temp=num,count=0,temp1,temp2;
9 while(temp>0){
10 temp1=temp%10;
11 count++;
12 temp/=10;
13 }
14 int i=count-1,num1=num;;
15 while(num1>0){
16 temp2=num1%10;
17 res+=temp2 * Math.pow(10,i);
18 num1/=10;
19 i-=1;
20 }
21 if(res==num)
22 System.out.println(num + " is a palindrome");
23 else
24 System.out.println(num + " is not a palindrome");
25 }
26 }
Terminal
Test cases

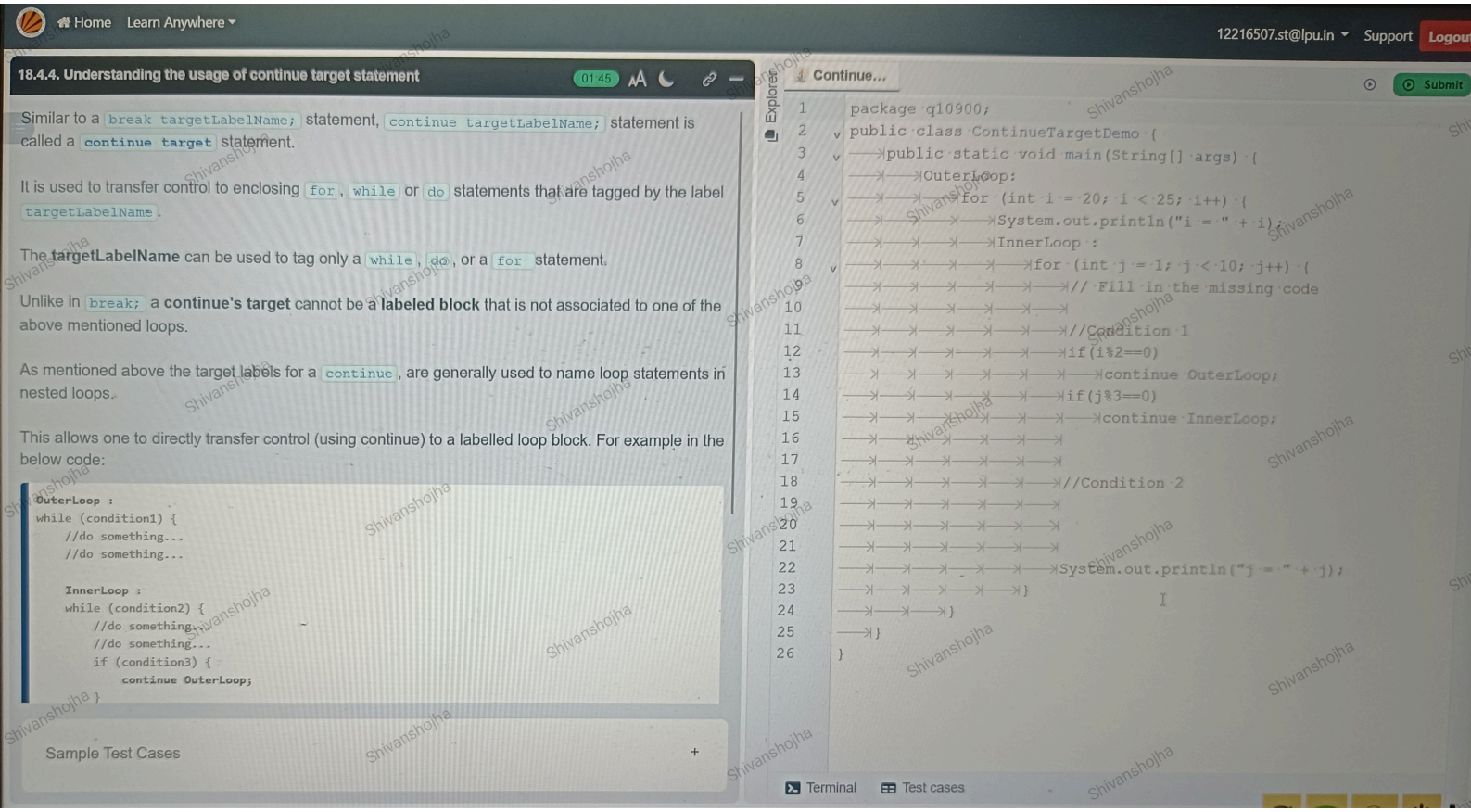

18.3.1. Understanding do-while loop statement
01:11
A do-while loop statement is used to execute one or more code statements once and then repeatedly execute the same code statements as long as the condition is true.
Its syntax is similar to the reverse of while loop, and note the extra ; which we have to provide at the end of while(condition)
do {
statement1;
statement2;
...
} while (condition);
Note: The statements inside the do-while block will be executed once before the condition in the while is evaluated.
The condition is an expression which should always evaluate to a boolean value.
• If it evaluates to true, the body containing one or more code statements is executed.
• If the expression evaluates to false, the body containing code statements is not executed.
For example:
int value = 4, sum = 0;
do { // start of do-while loop body
sum = sum + value;
value--;
} while (value > 0);// end of do-while loop body
1. The condition in the above example is value > 0.
2. The code inside the do-while loop's block between the opening-brace { and the closing-brace } , will be repeatedly executed until the condition evaluates to false.
3. In our case, since in every iteration the value is decremented using value--; , after four iterations the value will be equal to 0 and the condition value > 0 will become false.
Sample Test Cases
+
Explorer
DoWhile...
DoWhileD...
Submit
1 package q10895;
2 public class DoWhileDemo {
3 public void printNumbers(int max) {
4 int num = 1;
5
6 // Fill in the missing code using do-while statement to print 1 to max
7
8 do{
9 System.out.println(num);
10 num++;
11 }while (num<=max);
12 }
13 }
Terminal
Test cases

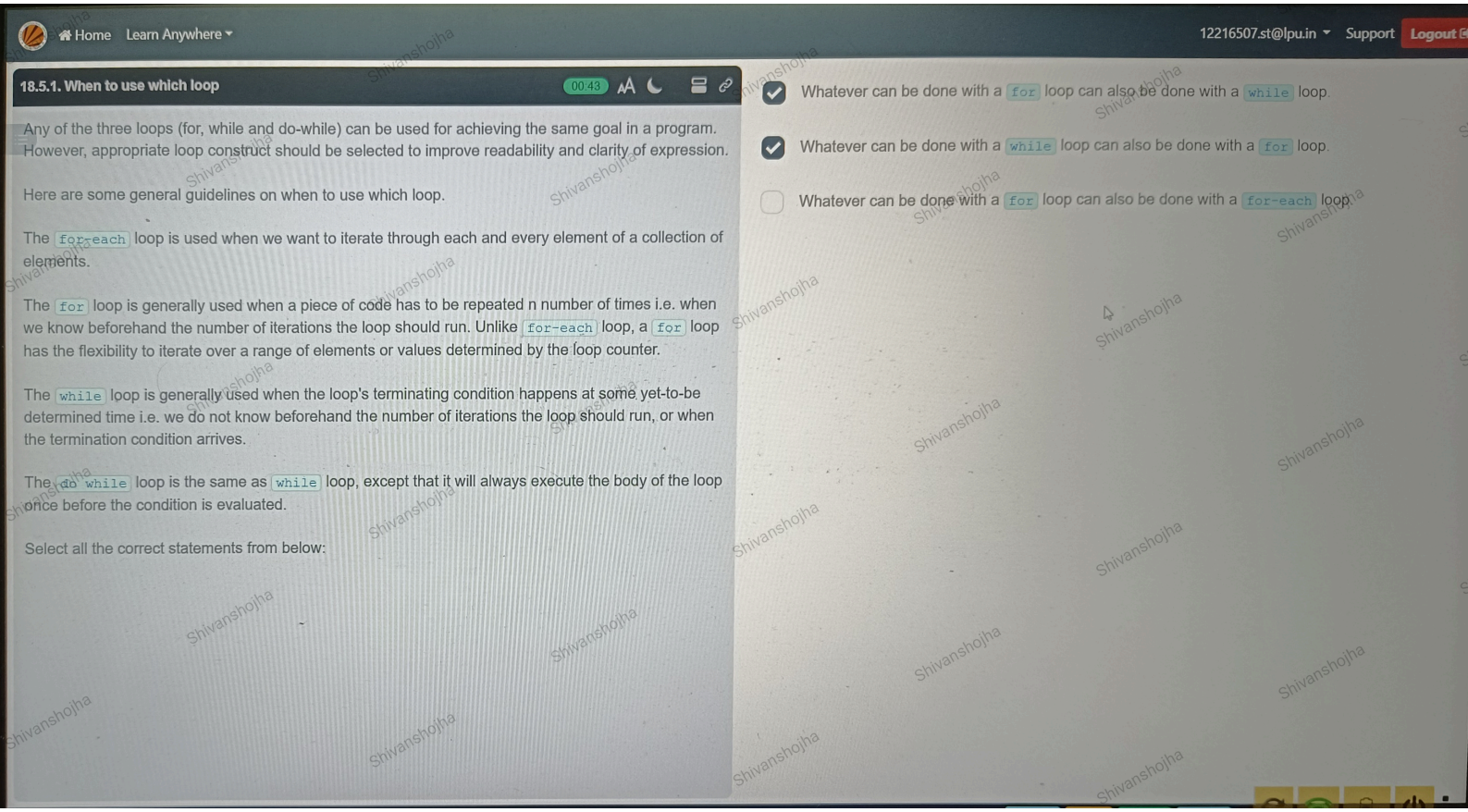

Home
Learn Anywhere
12216507.st@lpu.in
Support
Logout
18.3.2. Java Program to Display the Fibonacci Series
16:19
Write a class FibonacciSeries with a main method. The method receives one command line argument. Write a program to display fibonacci series i.e. 0 1 1 2 3 5 8 13 21.....
For example:
Cmd Args : 80
0 1 1 2 3 5 8 13 21 34 55
Note: Please don't change the package name.
Sample Test Cases
Explorer
Fibonacci...
Submit
1 package q10896;
2 public class FibonacciSeries{
3 public static void main(String args[]){
4 int num=Integer.parseInt(args[0]);
5 int a=0,b=1,res=1;
6 System.out.print(a);
7 while(res<=num){
8 System.out.print(" "+res);
9 res=a+b;
10 a=b;
11 b=res;
12
13 }
14 }
15 }
Terminal
Test cases


Home
Learn Anywhere
12216507.st@lpu.in
Support
Logout
18.4.1. Understanding the usage of break statement
03:32
A break; statement is used to transfer control out of the enclosing switch, for, while and do-while statements.
A break statement can be written simply as break; or with a label as break targetLabelName;
for (int i = 1; i < 10; i++) {
if (i % 5 == 0) {
break; // breaks from for-loop, if i is divisible by 5.
}
System.out.println(i);
}
Click on Live Demo to understand the working of break; statement.
Write a class BreakDemo with a public method calculateSum that takes one parameter arr of type int and returns the sum of all elements, untill it encounters a negative number. The return type of calculateSum should be int.
During the iteration if the code encounters a negative integer, the code should break (stop) from the iteration and return the sum of integers it encountered till then.
For example:
Cmd Args : 1 2 3 -4 5
6
Fill the missing code in the below program.
Sample Test Cases
Explorer
BreakDe...
BreakDe...
Submit
package q10897;
public class BreakDemo {
/**
* Calculate sum of numbers till the -ve number occures
*
*
* @return sum
*/
public int calculateSum(int[] arr) {
//Write your code here
int sum=0;
for(int i:arr){
if(i>=0)
sum+=i;
else
break;
}
return sum;
}
}
Terminal
Test cases
Prev
Reset
Submit
Next


Home
Learn Anywhere
12216507.st@lpu.in
Support
Logout
18.4.2. Understanding the usage of break target statement
03:08
A break targetLabelName; statement is called a break target statement.
It is used to transfer control out of the enclosing switch, for, while and do-while statements to the next statement below the statement labelled targetLabelName.
A break target need not always be a switch, while, do-while, or for statement, it could also be a simple block of statements enclosed in opening and closing braces { }, as shown in below example:
for (int i = 1; i < 10; i++) {
System.out.println("Step : 1");
SkipMe : {
if (i % 5 == 0) {
break SkipMe; // breaks to the SkipMe : label
}
System.out.println(i);
}
System.out.println("Step : 2");
}
The SkipMe : is called a label. A label is used to name a block of code, so that it can be targeted using a break and continue statements. Generally labels are used to name loop statements in nested loops, so that one can directly transfer control out of the labelled loop block. For example:
OuterLoop :
while (condition1) {
//do something...
/do something...
Sample Test Cases
Explorer
BreakTar...
Submit
Plots
package q10898;
public class BreakTargetDemo {
public static void main(String[] args) {
OuterLoop:
for (int i = 20; i < 25; i++) {
System.out.println("i = " + i);
InnerLoop :
for (int j = 1; j < 10; j++) {
// Write code to break OuterLoop, when i is a multiple of 11
if(i%11==0)
break OuterLoop;
if(j%5==0)
break InnerLoop;
// Write code to break InnerLoop, when j is a multiple of 5
System.out.println("j = " + j);
}
}
}
Terminal
Test cases


Home
Learn Anywhere
12216507.st@lpu.in
Support
Logout
18.4.3. Understanding the usage of continue statement
01:46
While a break; transfers control out (meaning terminates), a continue; statement transfers control to (meaning continues with next iteration) the innermost enclosing for, while and do statements.
Similar to a break statement, a continue statement can be written simply as continue; or with a label as continue targetLabelName;
for (int i = 1; i < 10; i++) {
if (i % 2 == 0) {
continue; // transfers control to for-loop, if i is divisible by 2.
}
System.out.println(i);
}
The above code skips printing all even numbers.
Click on Live Demo to understand the working of continue in action.
In the below program the method skipEven(int[] args) receives an array of integers.
Fill in the missing code inside the skipEven(...) such that it uses a for-each loop to iterate over the array arr elements and uses the continue; statement to skip even numbers.
Note: Please don't change the package name.
Sample Test Cases
Explorer
Continue...
Continue...
Submit
Plots
1 package q10899;
2 public class ContinueDemo {
3 public void skipEven(int[] arr) {
4 // Fill in the missing code
5 for (int i:arr) {
6 if(i%2==0)
7 continue;
8 else
9 System.out.println(i);
10 }
11 }
12 }
Terminal
Test cases


Home
Learn Anywhere
12216507.st@lpu.in
Support
Logou
18.4.4. Understanding the usage of continue target statement
01:45
Similar to a break targetLabelName; statement, continue targetLabelName; statement is called a continue target statement.
It is used to transfer control to enclosing for, while or do statements that are tagged by the label targetLabelName.
The targetLabelName can be used to tag only a while, do, or a for statement.
Unlike in break; a continue's target cannot be a labeled block that is not associated to one of the above mentioned loops.
As mentioned above the target labels for a continue, are generally used to name loop statements in nested loops.
This allows one to directly transfer control (using continue) to a labelled loop block. For example in the below code:
OuterLoop :
while (condition1) {
//do something...
//do something...
InnerLoop :
while (condition2) {
//do something...
//do something...
if (condition3) {
continue OuterLoop;
}
Sample Test Cases
Explorer
Continue...
Submit
package q10900;
public class ContinueTargetDemo {
public static void main(String[] args) {
OuterLoop:
for (int i = 20; i < 25; i++) {
System.out.println("i = " + i);
InnerLoop :
for (int j = 1; j < 10; j++) {
// Fill in the missing code
//Condition 1
if(i%2==0)
continue OuterLoop;
if(j%3==0)
continue InnerLoop;
//Condition 2
System.out.println("j = " + j);
}
}
}
Terminal
Test cases


Home
Learn Anywhere
12216507.st@lpu.in
Support
Logout
18.5.1. When to use which loop
00:43
Any of the three loops (for, while and do-while) can be used for achieving the same goal in a program. However, appropriate loop construct should be selected to improve readability and clarity of expression.
Here are some general guidelines on when to use which loop.
The for-each loop is used when we want to iterate through each and every element of a collection of elements.
The for loop is generally used when a piece of code has to be repeated n number of times i.e. when we know beforehand the number of iterations the loop should run. Unlike for-each loop, a for loop has the flexibility to iterate over a range of elements or values determined by the loop counter.
The while loop is generally used when the loop's terminating condition happens at some yet-to-be determined time i.e. we do not know beforehand the number of iterations the loop should run, or when the termination condition arrives.
The do while loop is the same as while loop, except that it will always execute the body of the loop once before the condition is evaluated.
Select all the correct statements from below:
Whatever can be done with a for loop can also be done with a while loop.
Whatever can be done with a while loop can also be done with a for loop.
Whatever can be done with a for loop can also be done with a for-each loop.

Home Learn Anywhere 12216507.st@lpu.in Support Logout

19.1.1. Introduction to Arrays

06:47

In a programming language the data that has to be processed is loaded in memory and held in variables. When we want to process an integer value, we declare an `int` variable and assign the value to it.
Suppose we want to write a program that prints the total marks scored by students in a class (say, for a total of 30 students).
One way of doing it is to declare 30 `int` variables, which is unwieldy when we think of multiple classes or a school.

In such cases, when we want to store multiple values of the same data type, we use an array data structure.
A `data structure` is a particular way of organizing data in a computer so that data can be accessed and modified efficiently.
An `array` is a kind of data structure that holds a fixed number of values of a single type, each identified by an `array index`.

For example, an array of size 10 (`int[] marksArr = new int[10];`) can be visualized as shown below.

0	1	2	3	4	5	6	7	8	9

Essentially an array can be thought of as a sequence of buckets. The first bucket is identified with number 0, the second bucket with 1 and so on. This number is called the `index`.

An element is stored in a bucket using the bucket's index. For example, if we want to store a value of 341 at the first index, the code is `marksArr[0] = 341;`.

Sample Test Cases +

Explorer ArrayDe... Submit

```
1  package q10935;
2  import java.util.*;
3  public class ArrayDemo {
4      public static void main(String[] args) {
5          int[] ramanujanNumbers = { 1729, 4104, 13832, 20683, 32832
   };
6          Scanner sc = new Scanner(System.in);
7          int n = sc.nextInt();
8
9          //Write your code here
10         System.out.println("Printing Ramanujan Numbers: ");
11         for(int i=0;i<n;i++){
12             System.out.println(ramanujanNumbers[i]);
13         }
14     }
15 }
```

Terminal Test cases

< Prev Reset Submit Next >

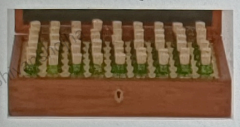
Home Learn Anywhere 12216507.st@lpu.in Support Logout

19.2.1. Understanding Multi-dimensional Arrays

01:35

In Java, a `multidimensional array` is implemented as an array of arrays.

A `multidimensional array` is an array whose components are themselves arrays. Imagine a medical cabinet like this:



The cabinet has multiple rows of medicines. Each row in the cabinet is an `array` and the whole cabinet is a `multidimensional array`.

In Java, the arrays / rows inside a `multidimensional array` can be of different lengths, like below -

0	1	2	3	4	5	6	7	8	9	→	Array at index 0

47	49	42	45	60	66	→	Array at index 1

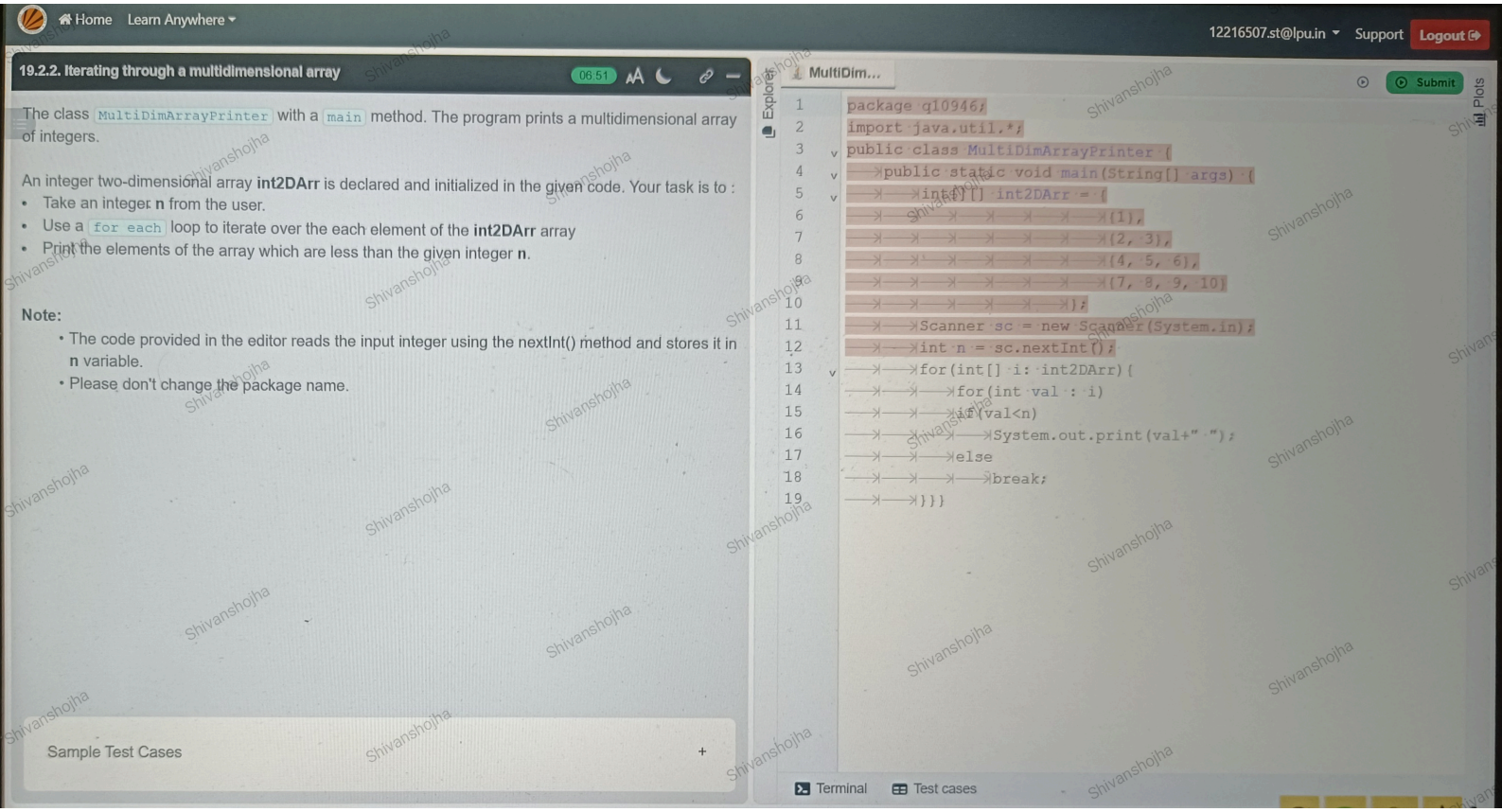
15	16	17	18	19	20	21	22	23	24	25	26	→	Array at index 2

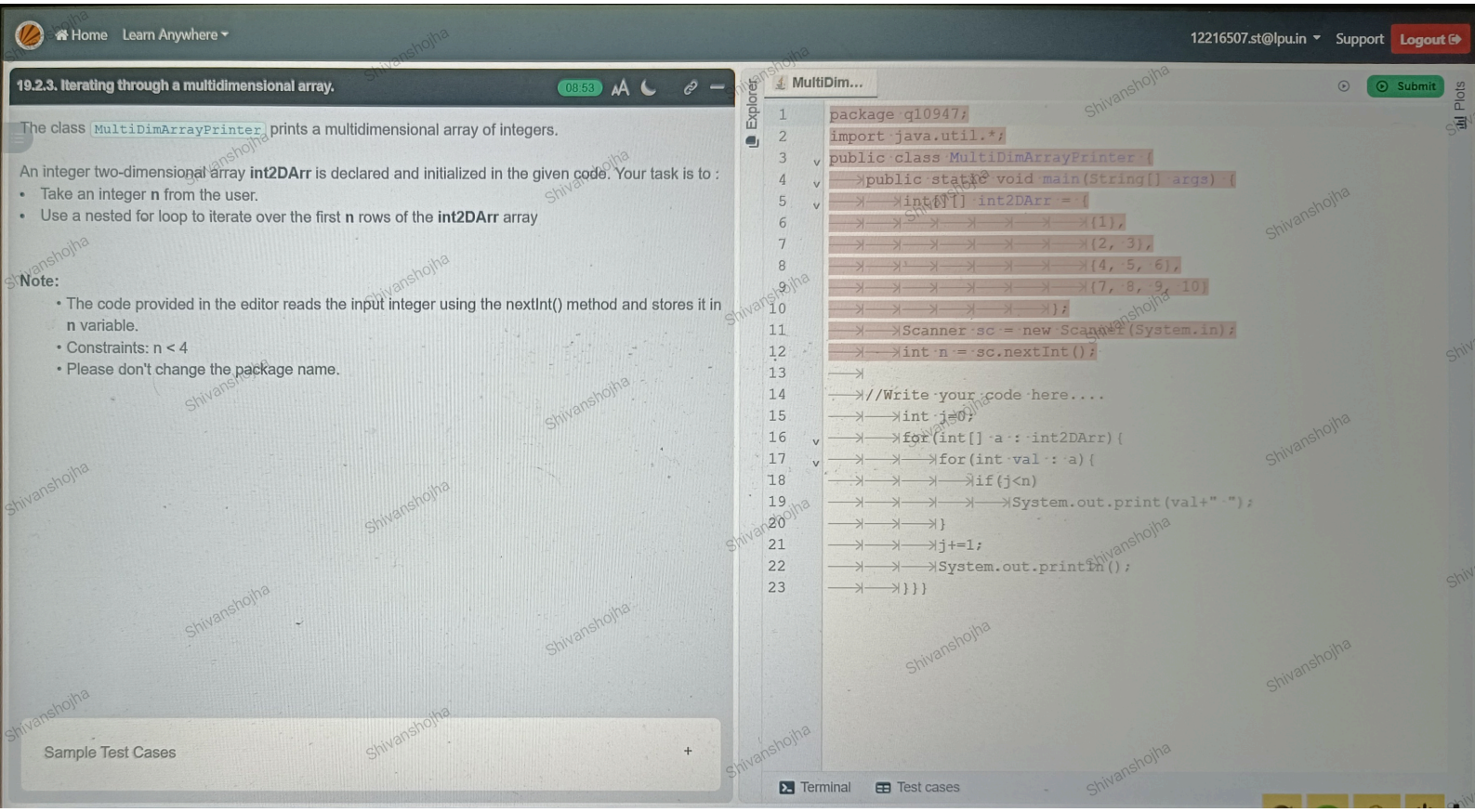
32	10	22	→	Array at index 3

Each `array` inside of a `multidimensional array` can be accessed with an `index` number as shown in the above figure. The first `array` is identified with number 0, the second `array` with 1 and

- [x] `int[][] intArray = new int[2][5];`
- [x]
```
int[][] intArray = new int[3][];
int[] arr1 = new int[5];
int[] arr2 = new int[6];
int[] arr3 = new int[8];

intArray[0] = arr1;
intArray[1] = arr2;
intArray[2] = arr3;
```
- [x]
```
int[] arr1 = new int[5];
int[] arr2 = new int[6];
int[] arr3 = new int[8];

int[][] intArray = {arr1, arr2, arr3};
```
- [x]
```
int[] arr1 = new int[5];

int[][] intArray = {arr1, {10,11,12}, {100,150,200}};
```
- [x] `int[][] intArray = {{10,11,12}};`
- [ ] `int[][] intArray = {{10.0,11,12}};`

< Prev Reset Submit Next >


Home
Learn Anywhere
12216507.st@lpu.in
Support
Logout
19.2.2. Iterating through a multidimensional array
06:51
The class MultiDimArrayPrinter with a main method. The program prints a multidimensional array of integers.
An integer two-dimensional array int2DArr is declared and initialized in the given code. Your task is to :
Take an integer n from the user.
Use a for each loop to iterate over the each element of the int2DArr array
Print the elements of the array which are less than the given integer n.
Note:
The code provided in the editor reads the input integer using the nextInt() method and stores it in n variable.
Please don't change the package name.
Sample Test Cases
MultiDim...
Submit
Explorer
Plots
1 package q10946;
2 import java.util.*;
3 public class MultiDimArrayPrinter {
4 public static void main(String[] args) {
5 int[][] int2DArr = {
6 {1},
7 {2, 3},
8 {4, 5, 6},
9 {7, 8, 9, 10}
10 };
11 Scanner sc = new Scanner(System.in);
12 int n = sc.nextInt();
13 for(int[] i: int2DArr){
14 for(int val : i)
15 if(val<n)
16 System.out.print(val+" ");
17 else
18 break;
19 }}}
Terminal
Test cases


Home
Learn Anywhere
12216507.st@lpu.in
Support
Logout
19.2.3. Iterating through a multidimensional array.
08:53
The class MultiDimArrayPrinter prints a multidimensional array of integers.
An integer two-dimensional array int2DArr is declared and initialized in the given code. Your task is to :
Take an integer n from the user.
Use a nested for loop to iterate over the first n rows of the int2DArr array
Note:
The code provided in the editor reads the input integer using the nextInt() method and stores it in n variable.
Constraints: n < 4
Please don't change the package name.
Sample Test Cases
MultiDim...
Submit
Explorer
Plots
1 package q10947;
2 import java.util.*;
3 public class MultiDimArrayPrinter {
4 public static void main(String[] args) {
5 int[][] int2DArr = {
6 {1},
7 {2, 3},
8 {4, 5, 6},
9 {7, 8, 9, 10}
10 };
11 Scanner sc = new Scanner(System.in);
12 int n = sc.nextInt();
13
14 //Write your code here....
15 int j=0;
16 for(int[] a : int2DArr){
17 for(int val : a){
18 if(j<n)
19 System.out.print(val+" ");
20 }
21 j+=1;
22 System.out.println();
23 }}}
Terminal
Test cases