Assignments (Misc.)

1. Consider the following code snippet:

**public** **class** ByteBuddyDemo {

**public** String createDynamicMethod() **throws** InstantiationException, IllegalAccessException {

Class<?> dynamicType = **new** ByteBuddy()

.subclass()//complete

.method() //complete

.intercept()// complete

.make()

.load()

.getLoaded();

**return** dynamicType.~~newInstance~~().toString();

}

}

Complete the code to create a Dynamic class and override toString() with output “Hi there”

1. Complete the following code:

**public** **class** ProcessStream {

**public** List<Integer> number;

**public** List<Integer> square;

**public** List<String> names = Arrays.*asList*("Sparrow","Cobra","Star",”Kite”);

**public** List<String> result;

**public** **void** process(){

// create a list of integers

number = Arrays.*asList*(3,2,5,6);

//write code to convert every number in the list to its square and collect them in a List

// Write code using stream and filter all the names start with "S"

// and store in arraylist in result.

// complete the code

}

}

1. Consider the following interface :

@FunctionalInterface

**public** **interface** Calculation {

**public** **int** getCal(**int** num1, **int** num2);

}

Complete the code in the following class:

**public** **class** ICalculation {

**int** number1=200,number2=2, add=0, sub=0, mul=0, div=0;

**public** **void** doCalculation(){

Calculation obj = **null**;

obj = //implement obj by adding lambda expression to add number1 and number2

add = obj.getCal(number1, number2);

obj = //implement obj by adding lambda expression to subtract number1 and number2

sub= obj.getCal(number1, number2);

obj = //implement obj by adding lambda expression to multiply number1 and number2

mul = obj.getCal(number1, number2);

obj = //implement obj by adding lambda expression to divide number1 and number2

div = obj.getCal(number1, number2);

}

}

1. In the following class:

**public** **class** PrimeTest {

**boolean** isPrime(**int** n)

{

// Corner case

**if** (n <= 1)

**return** **false**;

// Check from 2 to square root of n

**for** (**int** i = 2; i <= Math.*sqrt*(n); i++)

**if** (n % i == 0)

**return** **false**;

**return** **true**;

}

}

// complete the below code for unit test case to check a number is Prime or not

// do not use @Test annotation

**public** **void** testcase1(int data) {

// use data for testing

// use assert fail to fail the test case

}

}

1. Consider the given template:

**public** **class** ThreadDemo {

List<Data> list = **new** Vector<>();

{

**for** (**int** i = 0; i < 20; i++) {

Dataclass d = **new** Dataclass();

d.setAge(i + 10);

d.setCity("city");

d.setName("name");

list.add(d);

}

}

**public** **void** runthreads() {

//create 2 Thread Objects, Thread1 updates the Vector from start index and thread 2 updates the //Vector from end index. Threads just change the city to Uppercase in Dataclass.

}

Thread class Template is given below. You need to override run() method to write your logic.

**public** **class** UpdateThread **extends** Thread {

**public** List<Dataclass> obj;

**public** **int** index;

**public** UpdateThread (List<Dataclass> obj, **int** index) {

**this**.obj = obj;

**this**.index = index;

}

// override run method

// the run method should convert city to upper case based on the given index

}