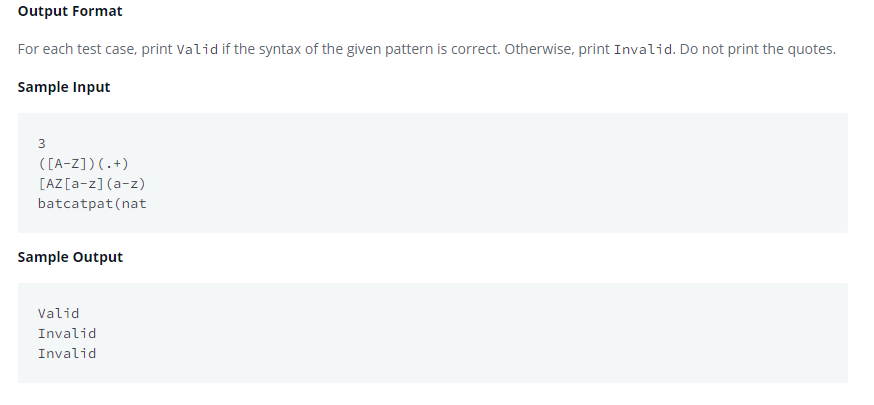
**NAMA : JAMALUDIN**

**TYPE : EASY**

**TUGAS HACKERRANKS DAY 3**

1. **Question “Pattern Syntax Checker”**



**Answer**

**import** java.util.Scanner;

**import** java.util.regex.\*;

**public** **class** Solution

{

**public** **static** **void** main(String[] args){

        Scanner in = **new** Scanner(System.in);

**int** testCases = Integer.parseInt(in.nextLine());

**while**(testCases>0){

            String pattern = in.nextLine();

**try** {

                Pattern.compile(pattern);

                System.out.println("Valid");

            } **catch** (PatternSyntaxException e) {

                System.out.println("Invalid");

            }

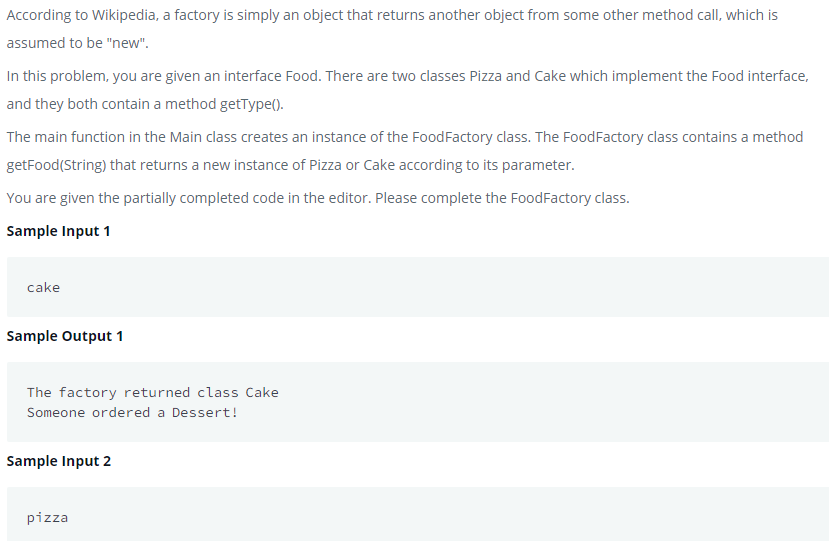
            testCases--;

        }

    }

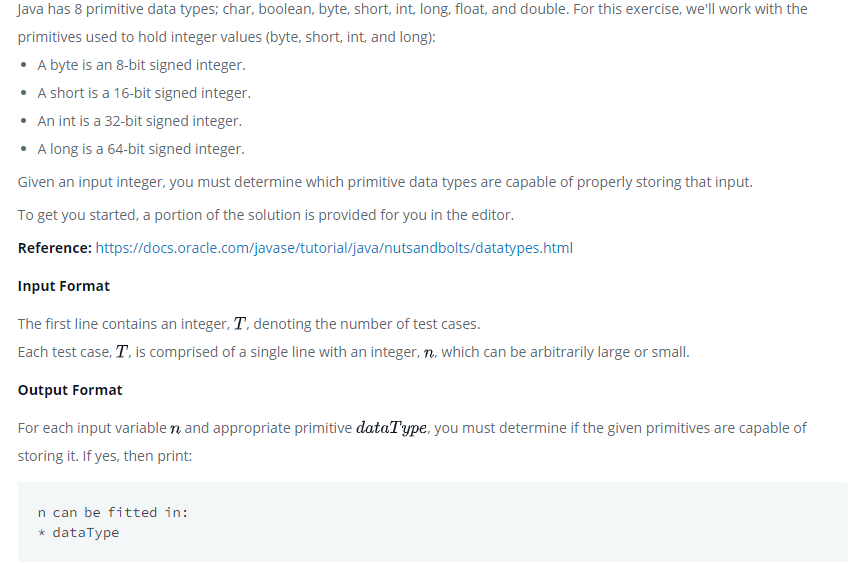
}

1. **“Java Factory”**



1. **import** java.util.\*;
2. **import** java.security.\*;
3. **interface** Food {
4. **public** String getType();
5. }
6. **class** Pizza **implements** Food {
7. **public** String getType() {
8. **return** "Someone ordered a Fast Food!";
9. }
10. }
11. **class** Cake **implements** Food {
12. **public** String getType() {
13. **return** "Someone ordered a Dessert!";
14. }
15. }
16. **class** FoodFactory {
17. **public** Food getFood(String order) {
18. **switch** (order){
19. **case** "pizza": **return** **new** Pizza();
20. **case** "cake" : **return** **new** Cake();
21. **default** : **return** null;
22. }
23. }*//End of getFood method*

**3. Question “Java DataTypes”**



**Answer**

**import** java.util.\*;

**import** java.io.\*;

**class** Solution{

**public** **static** **void** main(String []argh)

    {

        Scanner sc = **new** Scanner(System.in);

**int** t=sc.nextInt();

**for**(**int** i=0;i<t;i++)

        {

**try**

            {

**long** x=sc.nextLong();

                System.out.println(x+" can be fitted in:");

**if**(x>=-128 && x<=127)System.out.println("\* byte");

**if**(x>=-(Math.pow(2,15)) && x<=(Math.pow(2,15)-1))

                System.out.println("\* short");

**if**(x>=-(Math.pow(2,31)) && x<=(Math.pow(2,31)-1))

                System.out.println("\* int");

**if**(x>=-(Math.pow(2,63)) && x<=(Math.pow(2,63)-1))

                System.out.println("\* long");

            }

**catch**(Exception e)

            {

                System.out.println(sc.next()+" can't be fitted anywhere.");

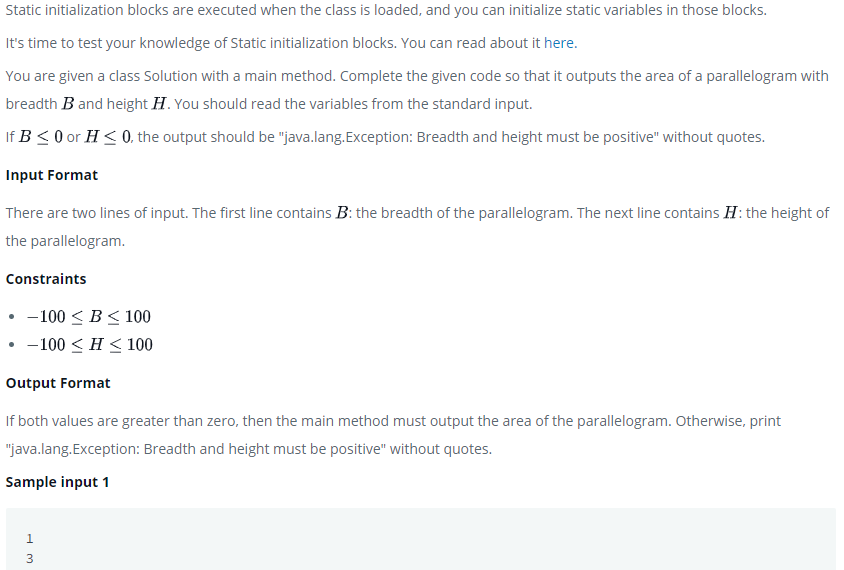
            }

        }

    }

}

**4. Question “Java Static Initializer Block”**



**Answer**

**import** java.io.\*;

**import** java.util.\*;

**import** java.text.\*;

**import** java.math.\*;

**import** java.util.regex.\*;

**public** **class** Solution {

**static** **boolean** flag = **true**; **static** **int** B,H;

**static**{

Scanner scan = **new** Scanner(System.in); B = scan.nextInt(); scan.nextLine(); H = scan.nextInt(); scan.close(); **if**(B>0 && H>0){ flag = **true**;

} **else** **if**((B<=0 && H>=0)||(B>=0 && H<=0)){ flag=**false**; System.out.println("java.lang.Exception: Breadth and height must be positive"); } **else** { flag=**false**; System.out.println("java.lang.Exception: Breadth and height must be positive"); }

}

**public** **static** **void** main(String[] args){

**if**(flag){

**int** area=B\*H;

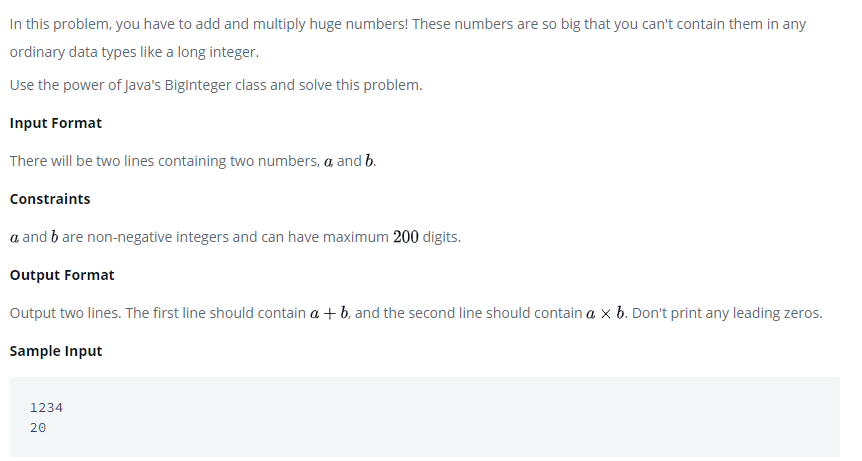
            System.out.print(area);

        }

    }*//end of main*

}*//end of class*

**5. Question “Java BigInteger”**



**Answer**

**import** java.io.\*;

**import** java.util.\*;

**import** java.text.\*;

**import** java.math.\*;

**import** java.util.regex.\*;

**public** **class** Solution {

**public** **static** **void** main(String[] args) {

         Scanner sc = **new** Scanner(System.in);

         BigInteger bi1 = **new** BigInteger(sc.next());

         BigInteger bi2 = **new** BigInteger(sc.next());

        BigInteger  bi3, bi4;

        bi3 = bi1.add(bi2);

        bi4 = bi1.multiply(bi2);

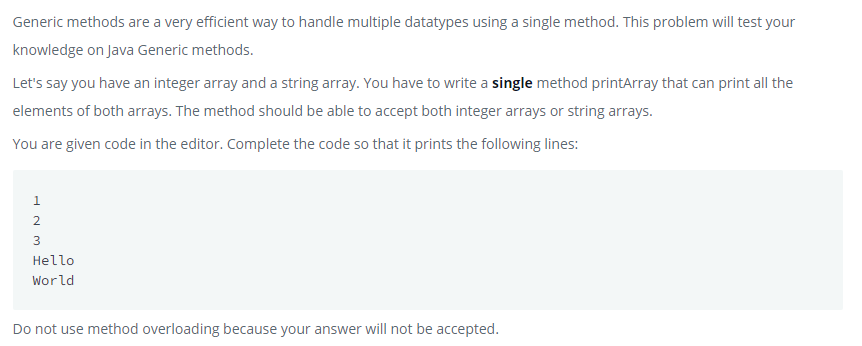
        System.out.println( bi3);

        System.out.println( bi4);

}

}

**6. Question “Java Generic”**



**Answer**

**import** java.io.IOException;

**import** java.lang.reflect.Method;

**class** Printer

{

**public** <T> **void** printArray(T[] array){

**for**(T item: array){

        System.out.println(item);

    }

}

}

**public** **class** Solution {

**public** **static** **void** main( String args[] ) {

        Printer myPrinter = **new** Printer();

        Integer[] intArray = { 1, 2, 3 };

        String[] stringArray = {"Hello", "World"};

        myPrinter.printArray(intArray);

        myPrinter.printArray(stringArray);

**int** count = 0;

**for** (Method method : Printer.**class**.getDeclaredMethods()) {

            String name = method.getName();

**if**(name.equals("printArray"))

                count++;

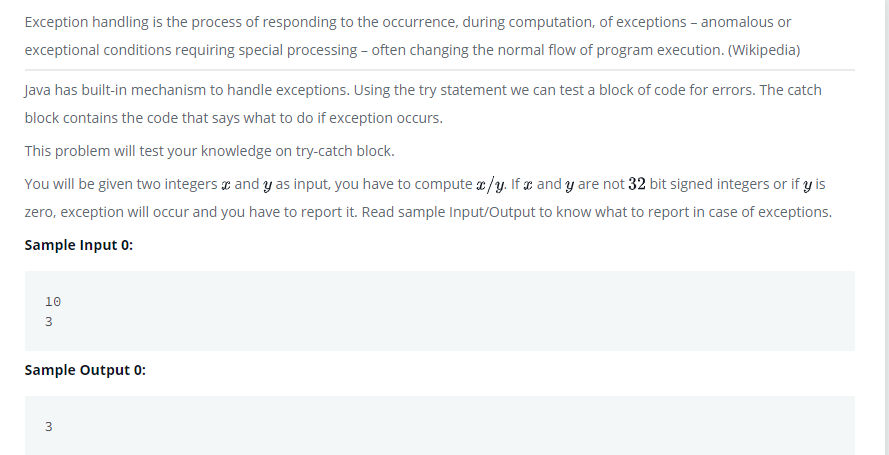
        }

**if**(count > 1)System.out.println("Method overloading is not allowed!");

    }

}

**7. Question “Java Exception Handling (Ttry-Catch)”**



**Answer**

**import** java.io.\*;

**import** java.util.\*;

**import** java.text.\*;

**import** java.math.\*;

**import** java.util.regex.\*;

**public** **class** Solution {

**public** **static** **void** main(String[] args) {

**try**{

            Scanner sc = **new** Scanner(System.in);

**int** x = sc.nextInt();

**int** y = sc.nextInt();

**if**(y==0)

**throw**  **new** ArithmeticException("/ by zero");

**else**

                System.out.println(x/y);

        }

**catch**(InputMismatchException e){

            System.out.println(e.getClass().getName());

        }

**catch**(ArithmeticException e){

            System.out.println(e);

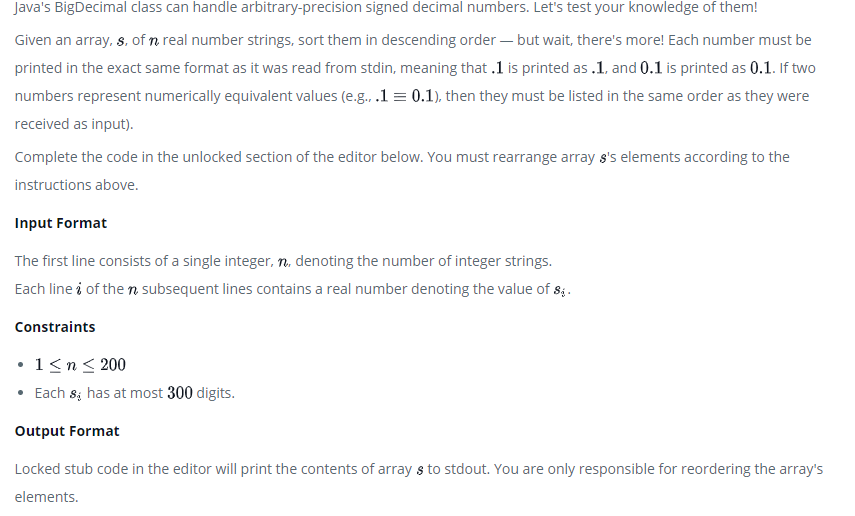
        }}

}

**NAMA : JAMALUDIN**

**TYPE : MEDIUM**

1. **Question “Java BigDecimal”**



**Answer**

**import** java.math.BigDecimal;

**import** java.util.\*;

**class** Solution{

**public** **static** **void** main(String []args){

*//Input*

        Scanner sc= **new** Scanner(System.in);

**int** n=sc.nextInt();

        String []s=**new** String[n+2];

**for**(**int** i=0;i<n;i++){

            s[i]=sc.next();

        }

        sc.close();

       Arrays.sort(s, **new** Comparator<String>() {

        @Override

**public** **int** compare(String o1, String o2) {

**if** (o1 == null || o2 == null) {

**return** 0;

        }

        BigDecimal o1bd = **new** BigDecimal(o1);

        BigDecimal o2bd = **new** BigDecimal(o2);

**return** o2bd.compareTo(o1bd);

        }

    });

*//Output*

**for**(**int** i=0;i<n;i++)

        {

            System.out.println(s[i]);

        }

    }

}