Code
for
Interviews
Java

W-13 2017-11-25

Day 1: ~~Hello World~~ Data types

diff: easy

```
import java.io.*;
import java.util.*;
import java.text.*;
import java.math.*;
import java.util.regex.*;

public class Day1 {
    public static void main(String[] args) {
        int i = 4;
        double d = 4.0;
        String s = "HackerRank";
        Scanner scan = new Scanner(System.in);
        int ii;
        double dd;
        String blank;
        String ss;
        ii = scan.nextInt();
        dd = scan.nextDouble();
        blank = scan.nextLine();
        ss = scan.nextLine();
        System.out.println(i+ii);
        System.out.println(d+dd);
        String sss = s+ss
        System.out.println(sss);
        scan.close();
    }
}
```

W13 2017-11-25
Day 2: Operations
Diff: easy

```
import java.util.*;
import java.[redacted].*;
Public class Day2 {
    Public Static void main(String[] args){
        Scanner Scan = new Scanner(System.in);
        double mealCost = Scan.nextDouble();
        int tipPercent = Scan.nextInt();
        int taxPercent = Scan.nextInt();
        [redacted].
        [redacted]
        Scan.close();
        double value = mealCost + mealCost *(tipPercent*0.01) +mealCost*(taxPercent*0.01);
        int totalCost =(int) Math.round(value);
        System.out.println("The total meal cost is "+ totalCost +" dollars.");
    }
}
```

W-13 2017-11-25
Day 3: Intro to Conditional Statements
Diff: easy

```
import java.io.*;
import java.util.*;
import java.text.*;
import java.math.*;
import java.util.regex.*;
Public Class Day3 {
    Public Static void main(String[] args){
        Scanner scan = new Scanner(System.in)
        int n = scan.nextInt();
        Scan.close();
        String.close();

        String ans = "";
        if(n%2==1){
            ans = "Weird";
        }
        else{
            if(n>=2 && n<=5){
                ans = "Weird";
            }
            if(n>=6 && n<=20){
                ans = "Weird";
            }
            if(n>20 && n<=100){
                ans= "Not Weird";
            }
        }
        System.out.println(ans);
    }
}
```

```
import java.io.*;
import java.util.*;
Public Class Person {
    private int age;
    Public Person (int initialAge ){
        if (initialAge >=0){
            age = initialAge;
        }
        else{
            age=0 ;
            System.out.println ("Age is not Valid,setting age to 0.");
        }
    }
    Public Void amIOld(){
        if(age >=0 && age<13){
            System.out.println("You are Young.");
        }
        else if (age >= 13 && age < 18){
            System.out.println (" You are a teenager");
        }
        else if (age >= 18){
            System.out.println ("You are old");
        }
    }
    Public Void yearPasses(){
        age ++;
    }
    Public Static void main (String[] args){
        Scanner sc = new Scanner(System.in);
        int T= sc.nextInt();
        for (int i=0 ; i<T; i ++){
            int age = sc.nextInt();
            Person p= new Person(age);
            p.amIOld();
            for (int j=0; j<3; j++){
                p.yearPasses();
            }
            p.amIOld
            System.out.println();
        }
        sc.close();
    }
}
```

Errors — Run code attempts 1111
Submit code 1

Day 4 Class vs. Instance

Input:

First line contains integer T,
and the subsequent T lines each
contain an integer denoting
the age of a Person instance.

Constraints:

$1 \le T \le 4$

$-5 \le age \le 30$

initial

W-13 2017-11-26

```
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 in = new Scanner(System.in);
        int n = in.nextInt();
        for(int i=1; i<=10; i++) {
            int value = n*i;
            System.out.println(n + " x " + i + " = " + value);
        }
    }
}
```

errors: none

Day # 5 - Loops

Input

- Single integer n

Constraints

$2 \leq n \leq 20$

W13 2017-11-26

```
import java.io.*;
import java.util.*;
import java.text.*;
import java.math.*;
import java.util.regex.*;
Public Class Solution {
    private String Str;
    Public Solution(String S);
        Str = S;
    }
    Public void even(){
        char[] charArr = this.Str.toCharArray();
        for(int i=0; i<this.Str.length(); i+=2){
            System.out.print(charArr[i]);
        }
    }
    Public void odd(){
        char[] charArr = this.Str.toCharArray();
        for(int i=1; i<this.Str.length(); i+=2){
            System.out.print(charArr[i]);
        }
    }
    Public Static void main(String[] args){
        Scanner sc = new Scanner(System.in);
        int t = sc.nextInt();
        String empty = sc.nextLine();
        for(int i=0; i<t; i++){
            String S = sc.nextLine();
            Solution sol = new Solution(S);
            sol.even();
            System.out.print(" ");
            sol.odd();
            System.out.println();
        }
        sc.close();
    }
}
```

Lets Review - Day 6 Errors Compile 111
Diff: easy RunCode
SubmitCode

Task - Given a String S of length N that is indexed from 0 to N-1, print its even-indexed and odd-indexed characters as 2 space-seperated Strings on a single line.

Input - first line contains an integer, T (number of Test cases)
- Each line i of the T subsequent lines contains a String, S.

Constraints
- $1 \leq T \leq 10$
- $2 \leq$ length of $S \leq 10000$

Output
each String S_j ($0 \leq j \leq T-1$), print S_j even indexed characters, followed by space, followed by S_j odd-indexed characters

W13 2017-11-26

Day 7: Arrays

2017-11-28

Difficulty: easy

Error: Compilation 11
Run code 1
Compilation 1
Run code 1

Task: given array A of N int, print A's elements in reverse order as a single line of space-separated numbers

Input: first line contains integer N (size of our array), second line contains N space-separated integers describing array A's elements

Constraints: $1 \leq N \leq 1000$
$1 \leq A_i \leq 10,000$ A_i is i^{th} int in array

Output: print elements of array A in reverse order as a single line of space-separated numbers.

```
import java.io.*
import java.util.*;
Public class Solution{
    Public Static int[] reverse(int size, int[] arr){
        for(int j=0; j<size; j++){
            int temp = arr[j];
            arr[j] = arr[size-1];
            arr[size-1] = temp;
            size --;
        }
        return (arr);
    }
    Public Static void printArr(int[] arr){
        for(i=0; i<arr.length; i++){
            System.out.print(arr[i] + " ");
    }
    }
    Public Static Void main(String[] args){
        Scanner in = new Scanner(System.in);
        int n = in.nextInt();
        int[] arr = new int[n];
        for(int i=0; i<n; i++){
            arr[i]=in.nextInt();
        }
        in.close();
        printArr(reverse(n,arr));
    }
}
```

// alternative using Foreach loop for printArr

```
Public Static Void printArr(int[] arr){
    for(int i:arr){
        System.out.print(i+" ");
    }
}
```

2017-11-28

Day 8: Dictionaries & Maps

Difficulty: easy

errors: Compile !!!!

```
import java.util.*;
import java.io.*;
class Solution {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        int n = in.nextInt();
        Map<String, Integer> phoneBook = new HashMap<String, Integer>();
        for(int i=0; i<n; i++){
            String name = in.next();
            int phone = in.nextInt();
            phoneBook.put(name, phone);
        }
        while(in.hasNext()) {
            String s = in.next();
            if(phoneBook.get(s)) != null){
                System.out.println(s + "=" + phoneBook.get(s));
            }
            else{
                System.out.println("Not found");
            }
            in.close();
        }
    }
}
```

→ ★ Integer is reference type

Task: Given n names and phone numbers, assemble a phoneBook that maps friends' names to their respective phone numbers. You will then be given an unknown number of names to query your phonebook for. For each name queried, print the associated entry from your phoneBook on a new line in the form name=phoneNumber if an entry for name is not found print Not found

Note: Your phone ~~xxxxxx~~ book should be a Dictionary/Map/Hashmap data structure

Input form: The first line contains an integer n, denoting the number of entries in the phone book.

Each of the n subsequent lines ~~describing~~ describes an entry in the form of 2 space-separated values on a single line. The first value is a friend's name, and the second value is an 8-digit phone number.

After the n lines of phone book entries, there are an unknown number of lines of queries. Each line (query) contains a name to look up, and you must continue reading lines until there is no more input.

Note: Names consist of lowercase English alphabetic letters and are first names only.

Constraints: $1 \leq n \leq 10^5$ $1 \leq \text{queries} \leq 10^5$

Output format: On a new line for each query, print Not found if the name has no corresponding entry in the phone book; otherwise, print the full name and phone number in the form name=phoneNumber

2017-11-28

Day 4 recursion error Compilation !
Difficulty: easy run code !

```
import java.io.*
import java.util.*;
import java.text.*;
import java.math.*;
import java.util.regex.*;
Public Class Solution {
    Static int factorial(int n){
        if(n>1){
            return n* factorial(n-1);
        }
        else {
            return n;
        }
    }
    Public Static void main(String[] args){
        Scanner in = new Scanner(System.in);
        int n= in.nextInt();
        int result = factorial(n);
        System.out.println(result);
    }
}
```

Task: Write a recursive function that takes a positive integer, N as a parameter and prints the result N!

Input: Single Integer, N

Constraints: $2 \leq N \leq 12$

Output: print a single integer denoting N!

Day 10: Binary Numbers

Compile ////
Test case //

2017-11-29

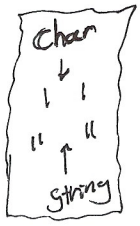
Reached max Int
→ 2147483647
default size of String builder

Difficulty: Easy

First attempt tried with only Int values

```
import java.io.*;
import java.util.*;
import java.text.*;
import java.math.*;
import java.util.regex;
Public class Solution {
    Public Static String decimalToBinaryString(int d) {
        int p=0;
        StringBuilder sb = new StringBuilder();
        While (true){
            if (d==0){
                breeak;
            }
            else {
                int temp= d%2
                Sb.append(temp);
                d= d/2
            }
        }
        StringBuilder sbr = sb.reverse();
        return (sbr.toString());
    }
    Public Static in maxConsecutiveOnes(String bString){
        Char[]b= bString.toCharArray();
        int count =0;
        int maxCount =0;
        for (int i=0 ; b.length ; i++){
            if(b[i]=='1'){
                count ++;
            }
            if (maxCount < count){
                maxCount = count;
            }
            if (b[i]=='0'){
                count =0;
            }
        }
        return(maxCount);
    }
    Public Static Void main (String[] args){
        Scanner sc = new Scanner(System.in);
        int n= sc.nextInt();
        System.out.println(maxConsecutiveOnes(decimalToBinaryString(n)));
        this. sc.close();
    }
}
```

attempt 2 With character array:
Compile ////



Task: Given a base-10 integer, n. Convert it to binary (base-2). Then find and print the base-10 integer denoting the maximum number of consecutive 1's in n's binary representation

Input Format
A single integer n;

Constraints
$1 \le n \le 10^6$

Output Format
Print a single base-10 integer denoting the maximum number of consecutive 1's in the binary representation of n.

```
import java.io.*;
import java.util.*;
import java.text.*;
import java.math.*;
import java.util.regex.*;
Public Static int calculateHourglass(int arr[][], int i, int j){
        return( arr[i][j]+ arr[i][j+1] + arr[i][j+2] +
                arr[i+1][j+1] +
                arr[i+2][j] + arr[i+2][j+1] + arr[i+2][j+2]
              );
}
Public Static int maxHourglassValue(int arr[][]){
    int maxValue= Integer.Min_Value;
    int value=0;
    for(int i=0; i<arr[0].length-2; i++){
        for(int j=0; j<arr[i].length-2; j++){
            value= calculateHourglass(arr, i, j);
            if(value > maxValue){
                maxValue = value
            }
        }
    }
    return(maxValue);
}
Public Static Void main(String[] args){
    Scanner in = new Scanner(System.in);
    int arr[][] = new int[6][6];
    for(int i=0; i<6; i++){
        for(int j=0; j<6; j++){
            arr[i][j] = in.nextInt();
        }
    }
    System.out.println(maxHourglassValue(arr));
    in.close();
}
```

Day 11: 2D Arrays
2017-11-30

errors: compile
test: 1
2 cases failed

did not account for negative value;
initial value should be Set for Integer.Min_Value

Task:
Calculate the hourglass Sum for every hourglass in A; then print the maximum hourglass Sum

Input Form
There are 6 lines of input, where each line contains 6 space-separated integer describing 2D Array A; every value in A will be in the inclusive range of -9 to 9.

Constraints:
$-9 \leq A[i][j] \leq 9$
$0 \leq i, j \leq 5$

Output:
Print the largest hourglass Sum found in A.

diff: easy
2017-11-30

Day 12: ~~Inheritance~~ errors compile: 11

inheritence

Task: Given two classes, Person and Student, where Person is the base class and Student is the derived class. Complete code for Person and a declaration for Student are provided for you in the editor. Observe that Student inherits all the properties of Person. Compile the Student class by writing the following: — A Student class

- constructor, which has 4 parameters:
  1. A String, firstName 2. A String, lastName
  3. An integer, id 4. An ~~array~~ integer array (or Vector) of test scores, scores.
- A char calculate() method that calculates a Student's object's average and returns the grade character ~~represented~~ representative of their calculated average:

O	$90 \le a \le 100$	T $a < 40$
E	$80 \le a < 90$	
A	$70 \le a < 80$	
P	$55 \le a < 70$	
D	$40 \le a < 55$	

Input format: locked stub code in your editor calls your Student class constructor and passes it the necessary arguments. It also calls the calculate method (which takes no arguments). You are not responsible for reading the following input from stdin. The first line contains firstName, lastName, and id. The second line contains the number of test scores. The third line of space-separated integers describes scores.

Constraints → $1 \le |\text{firstName}|, |\text{lastName}| \le 10$

→ $|id| = 7$ → $0 \le \text{score, average} \le 100$

Output format: This is handled by the locked stub code in your editor. Your output will be correct if your Student class constructor and calculate() method are properly implemented.

```
import java.util.*;
Class Person {
    protected String firstName;
    protected String lastName;
    protected int idNumber;
    Person(String firstName, String lastName, int identification) {
        this.firstName = firstName;
        this.lastName = lastName;
        this.idNumber = identification;
    }
    Public Void printPerson() {
        System.out.println(
            "Name: " + lastName + ", " + firstName
            + "\n ID: " + idNumber);
    }
}
Class Student extends Person {
    private int[] testScores;
    Student(String firstName, String lastName, int identification, int[] testScores) {
        super(firstName, lastName, identification);
        this.testScores = testScores;
    }
    char calculate() {
        int sum = 0;
        for(int i = 0; i < this.testScores.length; i++) {
            sum += this.testScores[i];
        }
        int avg = sum / this.testScores.length;
        if(avg >= 90 && avg <= 100) {
            return('O');
        }
        if(avg >= 80 && avg < 90) {
            return('E');
        }
        if(avg >= 70 && avg < 80) {
            return('A');
        }
        if(avg >= 55 && avg < 70) {
            return('P');
        }
        if(avg >= 40 && avg < 55) {
            return('D');
        }
        if(avg < 40) {
            return('T');
        }
        else {
            System.out.println("error");
            return('O');
        }
    }
}
```

```
Public Class Solution {
    public static Void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        String firstName = scan.next();
        String lastName = scan.next();
        int id = scan.nextInt();
        int numScores = scan.nextInt();
        int[] testScores = new int[numScores];
        for(int i = 0; i < numScores; i++) {
            testScores[i] = scan.nextInt();
        }
        scan.close();
        Student s = new Student(firstName,
                                lastName,
                                id,
                                testScores);
        s.printPerson();
        System.out.println("Grade: " + s.calculate());
    }
}
```

```
import java.util.*;
abstract class Book {
    String title;
    String author;
    Book(String title, String author) {
        this.title = title;
        this.author = author;
    }
    abstract void display();
}
class MYBook extends Book {
    int price;
    MYBook(String title, String author, int price) {
        super(title, author);
        this.price = price;
    }
    void display() {
        System.out.println("Title: " + this.title + "\n" +
                           "Author: " + this.author + "\n" +
                           "Price: " + this.price);
    }
}
public class Solution {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        String title = scanner.nextLine();
        String author = scanner.nextLine();
        int price = scanner.nextInt();
        scanner.close();
        Book book = new MYBook(title, author, price);
        book.display();
    }
}
```

Day 13: Abstract Classes errors compile: none ☆ !!

2017-11-30

Task: given a Book class and a Solution class. Write a MYBook class that does the following:

→ Inherits from Book

→ Has a parameterized constructor taking these 3 parameters: 1. String title 2. String author 3. int price

→ Implements the Book class abstract display() method so it prints these 3 lines:

1. Title:, a space, and then the current instance's title.
2. Author:, a space, and then the currents instance's author.
3. price, a space, and then the currents instance's price.

Note: Because these classes are being written in the same file, you must not use an access modifier (i.e. public) when declaring MYBook or your code will not execute.

Input Format

You are not responsible for reading any input from stdin. The Solution class creates a Book object and calls the MYBook class constructor (passing it the necessary arguments) It then calls the display method on the Book object.

Output Format

The void display() method should print and label the respective title, author, and price of MYBook objects instance (with each value on its own line)

```
import java.io.*;
import java.util.*;
import java.text.*;
import java.math.*;
import java.util.regex.*;
class Difference {
    private int[] elements;
    public int maximumDifference;
    Difference(int[] arr){
        this.elements = arr;
    }
    Void computeDifference(){
        int difference = 0;
        this.maximumDifference = 0;
        for(int i=0; i<this.elements.length; i++){
            for(int j=i+1; j<this.elements.length; j++){
                difference = Math.abs(this.elements[i] - this.elements[j]);
                if(difference > this.maximumDifference){
                    this.maximumDifference = difference;
                }
            }
        }
    }
}
Public class Solution {
    Public Static Void main(String[] args){
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        int[] a = new int[n]
        for(int i=0; i<n; i++){
            a[i] = sc.nextInt();
        }
        sc.close();
        Difference difference = new Difference(a);
        difference.computeDifference();
        System.out.print(difference.maximumDifference);
    }
}
```

Day 14 Scope errors: Compile Run no errors

Diff: Easy

Task: Complete the difference class by writing the following:

- A class constructor that takes an array of integers as a parameter and saves it to the elements instance variable.
- A computeDifference method that finds the maximum absolute difference between any 2 numbers in N and stores it in the maximumDifference instance variable

Input Form: You are not responsible for reading any input from stdin. The locked Solution class in your editor reads in 2 lines of input; the first line contains N, and the second line describes the elements in the array.

Constraints: $1 \leq N \leq 10$
$1 \leq elements[i] \leq 100$, where $0 \leq i \leq N-1$

Output Format:

You are not responsible for printing any output the Solution class will print the value of the maximumDifference instance variable.

```
import java.io.*;
import java.util.*;
class Node{
    int data;
    Node next;
    Node(int d){
        data=d;
        next=null;
    }
}
class Solution
    public static Node insert(Node head, int data){
        Node temp=new Node(data);
        if(head==null){
            head=temp;
            return(head);
        }
        else{
            Node current=head;
            while(current.next!=null){
                current=current.next;
            }
            current.next=temp;
            return(head);
        }
    }
    public static void display(Node head){
        Node start=head;
        while(start!=null){
            System.out.print(start.data+" ");
            start=start.next;
        }
    }
    public static void main(String args[]){
        Scanner sc=new Scanner(System.in);
        Node head=null;
        int N=sc.nextInt();
        while(N--->0){
            int ele=sc.nextInt();
            head=insert(head,ele);
        }
        display(head);
        sc.close();
    }
}
```

Day 15: Linked List : compile !!!
errors : Run
Null pointer exception

Task

Complete the insert function in your editor so that it creates a new Node (pass data as the Node constructor argument) and inserts it at the tail of the linked list referenced by the head parameter. Once the new node is added, return the reference to the head node.

Note: If the head argument passed to the insert function is null, then the initial list is empty

Input Format

- The insert function has 2 parameters: a pointer to a Node named head, and an integer value, data.

The constructor for Node has 1 parameter: an integer value for the data field. You do not need to read anything from Stdin

Output Format

Your insert function should return a reference to the head node of the linked list.

Day 16: Exceptions - String to Integer
Difficulty: ecesy
errors: Compile H

```
import java.io.*;
import.java.util.*;
import java.text.*;
import java.math.*;
import java.util.regex.*;
Public class Solution
    Public Static Void main (String[] cargs) {
        Scanner in = new Scanner (System.in);
        String S= in.next();
        try {
            int i = Integer.parseInt(S)
            System.out.println(i);
        } Catch (Exception e){
            System.out.println("Bad String");
        }
    }
}
```

Task Read string S, and print its integer value; if S cannot be converted to an integer, print "Bad String"

Note: You must use the string to integer and exception handling constructs built into your ~~Programming~~ Submission language. If you attempt to use loops/conditional statements, You will get a score of 0.

Input: a single string S

Constraints $1 \leq |S| \leq 6$
$|S|$ is the length of the string.
S is composed of either lowercase letters (a-z) or decimal digits (0-9)

Output Format
~~print the parsed integer value~~ of S, or Bad String if S cannot be converted to an integer

Difficulty: Easy

Task

Write a Calculator class with a single method: int power(int, int). The power method takes two integers, n and p, as parameters and returns the integer result of n^p. If either n or p is negative, then the method must throw an exception with the message "n and p should be non-negative".

Input Format

Input from stdin is handled for you by locked stub code in your editor. The first line contains an integer, T, the number of test cases. Each of the T subsequent lines describes a test case in 2 space-separated integers denoting n and p, respectively.

Constraints

No test case will overflow for correctly written code

Output Format

Output to stdout is handled by stub code. T lines of output, where each line contains the result of n^p as calculated by Calculator class power method.

```
import java.util.*;
import java.io.*;
class Calculator {
    int power(int n, int p) throws Exception {
        if (n<0 || p<0) {
            throw new Exception("n and p should be non-negative");
        }
        return (int)Math.pow(n,p);
    }
}
class Solution {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        int t = in.nextInt();
        while (t--)>0) {
            int n = in.nextInt();
            int p = in.nextInt();
            Calculator myCalculator = new Calculator();
            try {
                int ans = myCalculator.power(n,p);
                System.out.println(ans);
            }
            catch (Exception e) {
                System.out.println(e.getMessage());
            }
        }
        in.close();
    }
}
```

Day 18: Queues and Stacks compile 1

```
import java.io.*;
import java.util.*;
Public class Solution {
    LinkedList queue;
    Stack<Character> stack;

    Solution() {
        queue = new LinkedList();
        stack = new Stack<>();
    }
    Void pushCharacter(char ch){
        stack.push(ch);
    }
    void enqueueCharacter( char ch) {
        queue.addLast(ch);
    }
    Char popCharacter(){
        return stack.pop();
    }
    char dequeueCharacter(){
        return (char) queue.remove();
    }
    Public static void main(String[] args){
        Scanner scan = new Scanner(System.in);
        String input = scan.nextLine();
        scan.close();
        Char[] s = input.toCharArray();
        Solution p = new Solution();
        for(char c : s) {
            p.pushCharacter(c);
            p.enqueueCharacter(c);
        }
        boolean isPalindrome = true;
        for(int i=0; i<s.length/2; i++){
            if(p.popCharacter() != p.dequeueCharacter()){
                isPalindrome = false;
                break;
            }
        }
        System.out.println("The word, " + input + ", is " + ((!isPalindrome)? "not
        a palindrome." : "a palindrome."));
    }
}
```

Task: To solve this challenge we must first take each character in S, enqueue it in a queue and also push that same character onto a stack. Once thats done We must dequeue the first character from the queue and pop the top character off of the stack, then compare the two characters to see if they are the same, as long as the characters match We continue dequeuing, popping, and comparing each character until our containers are empty. (a non-match means S isn't a palindrome

Write the following declarations and implementations

1. Two instance variables: one for Stack and one for queue.
2. A void pushCharacter(char ch) that pushes a character in the queue instance variable
3. A void enqueueCharacter(char ch) that enqueues a character in the queue instance variable
4. popCharacter method that pops and returns the character at the top of the Stack Instance Variable
5. A char dequeue method that dequeues and returns the first character in the queue instance variable

Input: ~~do not need~~ - read from Stdin in main
- read single line string
- then call method to push each character to your instance variables

Constraints - S is composed of lower case english letters

~~Output Format~~ - code prints is a palindrome if S is a palindrome & not a palindrome if not a palindrome

Output Format

Note: Solution.java uses ~~unsafe~~ unchecked or unsafe operations

```
import java.io.*;
import java.util.*;
interface AdvancedArithmetic {
   int divisorSum(int n);
}
class Calculator implements AdvancedArithmetic {
   public int divisorSum(int n) {
      int sum = 0;
      for(int i = 1; i <= n; i++) {
         if (n % i == 0) {
            sum += i;
         }
      }
      return(sum);
   }
}
class Solution {
   public static void main(String[] args) {
      Scanner scan = new Scanner(System.in);
      int n = scan.nextInt();
      scan.close();
      AdvancedArithmetic myCalculator = new Calculator();
      int sum = myCalculator.divisorSum(n);
      System.out.println("I implemented: " + myCalculator.getClass().getInterfaces()[0]);
      System.out.println(sum);
   }
}
```

Day 19: Interfaces Compile error 11

~~Diff: easy~~

Diff: easy

Task: AdvancedArithmetic interface and method declaration for the abstract int divisorSum(int n) method are provided for you in the editor below. Write the Calculator class, which implements the AdvancedArithmetic interface. The implementation for the divisorSum method must be public and take an integer parameter, n, and return the sum of all its divisors. Because multiple classes in same file, do not use an access modifier (public) in class declaration, however,

1. You must use the public access modifier before your method declaration for it to be accessible by any other class in the file.

Input

Single line containing an integer, n.

Constraints

$1 \leq n \leq 1000$

Output

Locked Solution class in editor will call your code and print the necessary output.

- getName()

Day 20: Sorting Compile issue 1
Diff: easy error

Failed test ~~case~~ 1111

Checked bubble Sort offline

online

```
import java.io.*;
import java.util.*;
import java.text.*;
import java.math.*;
import java.util.regex.*;
Public class Solution {
    int numberOfSwaps;
    int [] array;
    Solution (int n, int [] a){
        this.numberOfSwaps = n;
        this.array = a;
    }
    Void bubbleSort() {
        int n = this.array.length;
        for(int i=0; i<= n-1; i++){
            for(int j=0; j< n-i-1; j++){
                if(this.array[j] > this.array[j+1]){
                    Swap(j, j+1);
                }
            }
        }
    }
    Void Swap(int i, int j){
        int temp = this.array[i];
        this.array[i] = this.array[j];
        this.array[j] = temp;
        this.numberOfSwaps ++;
    }
    int getNumberOfSwaps(){
        return(this.numberOfSwaps);
    }
    int getFirstElement(){
        return(this.array[0]);
    }
    int getLastElement(){
        return (this.array[this.array.length-1]);
    }
}
```

Task: ~~Write a Simple~~ Given an array, a, of size n distinct elements, sort the array in ascending ~~order~~ order using the bubble Sort algorithm.

Input Format: first line contains an integer, n, denoting number of elements in array a. The second line contains n space seperated integers describing the respective values of $a_0, a_1, a_2, \ldots a_{n-1}$

Constraints: $2 \le n \le 600$
$1 \le a_i \le 2 \times 10^6$ where $0 \le i < n$

Output Format
Print ~~after~~ Sorting complete
Array is sorted in "numSwaps" swaps.
First Element: "first Element"
Last Element: "the last Element"

```
Public Static Void main(String[] args){
    Scanner in = new Scanner(System.in);
    int n= in.nextInt();
    int[] a= new int[n];
    for(int a_i=0; a_i<n; a_i++){
        a[a_i] = in.nextInt();
    }
    Solution Sol= new Solution(0, a);
    Sol.bubbleSort();
    System.out.println("Array is Sorted in "+
                        Sol.getNumberOfSwaps();
                        + " swaps");
    System.out.println("First Element: " + Sol.getFirstElement());
    System.out.println("Last Element: "
                        + Sol.getLastElement());
}
```

"main"

2017-12-11
Diff: Easy

Day 21: Generics no errors

Task: Write a single generic function named printArray; this function must take an array of generic elements as a parameter.

Input Format: The locked Solution class in your editor will pass different types of array to your printArray function.

Constraints: You must have exactly 1 function named printArray

Output: Your printArray function should print each element of its generic array parameter on a new line.

```
import java.util.*;
class Printer<T>{
    public static <E> void printArray(E[] array){
        for(E element: array){
            System.out.println(element);
        }
    }
}
public class Generics{
    public static void main(String args[]){
        Scanner scanner = new Scanner(System.in);
        int n = scanner.nextInt();
        Integer[] intArray = new Integer[n];
        for(int i=0; i<n; i++){
            intArray[i]= scanner.nextInt();
        }
        n= scanner.nextInt();
        String[] stringArray = new String[n];
        for(int i=0; i<n; i++){
            stringArray[i]= scanner.next();
        }
        Printer<Integer> intPrinter = new Printer<Integer>();
        Printer<String> stringPrinter = new Printer<String>();
        intPrinter.printArray(intArray);
        stringPrinter.printArray(stringArray);
        if(Printer.class.getDeclaredMethods().length > 1){
            System.out.println("The printer class should only have 1 method named printArray.");
        }
    }
}
```

Day 22: Binary Search Tree Compile error: 11
Diff: easy. Testcase: 11

Had to look online about information for BST

Task: find height of a BST, given a pointer root, pointing to the root of the Binary Search tree. Complete the get height function

Input format: locked stub code, first line contains integer n, denoting number of nodes in the tree. each subsequent n lines contain an integer data denoting the value of the elements that must be added to the binary search tree

Output format: locked stub code prints value to be returned by the getHeight function

```
import java.util.*;
import java.io.*;
class Node {
    Node left, right;
    int data;
    Node(int data) {
        this.data = data;
        left = right = null;
    }
}
class Solution {
    public static int getHeight(Node root) {
        if(root = null) {
            return -1;
        }
        else {
            int lDepth = getHeight(root.left);
            int rDepth = getHeight(root.right);
            if(lDepth > rDepth) {
                return(lDepth + 1);
            }
            else {
                return(rDepth + 1);
            }
        }
    }
    public static Node insert(Node root, int data) {
        if(root == null) {
            return new Node(data);
        }
        else {
            Node cur;
            if(data <= root.data) {
                cur = insert(root.left, data);
                root.left = cur;
            }
            else {
                cur = insert(root.right, data);
                root.right = cur;
            }
            return root;
        }
    }
    public static void main(String args[]) {
        Scanner sc = new Scanner(System.in);
        int T = sc.nextInt();
        Node root = null;
        while(T-->0) {
            int data = sc.nextInt();
            root = insert(root, data);
        }
        int height = getHeight(root);
        System.out.println(height);
    }
}
```

"main"

Day 23: BST Level-Order Traversal

Difficulty: Easy

Looked at BST Level-Order Traversal on geeksforgeeks.org

Complile Error: IIII

TestCase Error: II

→ Poll() removes the present head from the queue.

Task: level order traversal, also known as breadth first search. Visit each of a trees nodes from top left to bottom right. given pointer, root pointing to the root of a binary search tree. Complete the level Order function provided in your editor so that it prints the level-order traversal of the binary search tree. Hint: queue is helpful when completing this challenge. Input: first line integer T, number of test cases. T subsequent lines contain integer data, denoting value of element to be added to the BST

Output print data value at each node in the trees level-order traversal as a single line of N space-seperated integers

```
import java.io.*;
import java.util.*;
class Node {
    Node left, right;
    int data;
    Node(int data) {
        this.data = data;
        left = right = null;
    }
}
class Solution {
    static void levelOrder(Node root) {
        Queue<Node> queue = new LinkedList<Node>();
        queue.add(root);
        while(!queue.isEmpty()) {
            Node tempNode = queue.poll();
            System.out.print(tempNode.data + " ");
            if(tempNode.left != null) {
                queue.add(tempNode.left);
            }
            if(tempNode.right != null) {
                queue.add(tempNode.right);
            }
        }
    }
    public static Node insert(Node root, int data) {
        if(root == null) {
            return new Node(data);
        }
        else {
            Node cur;
            if(data <= root.data)
                cur = insert(root.left, data);
                root.left = cur;
            else {
                cur = insert(root.right, data);
                root.right = cur;
            }
            return root;
        }
    }
    public static void main(String args[]) {
        Scanner sc = new Scanner(System.in);
        int T = sc.nextInt();
        Node root = null;
        while(T-->0) {
            int data = sc.nextInt();
            root = insert(root, data);
        }
        levelOrder(root);
    }
}
```

2018-01-20 Day 24: More Linked Lists Compile: no data
run: no data

```
import java.io.*;
import java.util.*;
Class Node {
    int data
    Node next;
    Node (int d) {
        data = d;
        next = null;
    }
}
Class Solution {
    Public ~~Node~~ static Node removeDuplicates (Node head) {
        HashSet<Integer> hs = new HashSet<>();
        Node current = head;
        Node prev = null;
        While (current != null) {
            int curval = current.data;
            if (hs.contains(curval)) {
                prev.next = current.next;
            } else {
                hs.~~add~~ add (curval);
                prev = current;
            }
            current = current.next;
        }
        return head;
    }
    Public static ~~Node~~ Node insert (Node head, int data);
    {
        Node p = new Node(data);
        if (head == null)
            head = p;
        else if (head.next == null)
            head.next = p;
        else {
            Node start = head;
            While (start.next != null) {
                start = start.next;
                ~~start~~
                start.next = p;
            }
        }
    Public static void display (Node head) {
        Node start = head;
        While (start != null)
        { System.out.print (start.data + " ");
        } start = start.next;
    }
}
```

Task:

Complete removeDuplicates function so that it deletes any duplicates nodes from the list and returns the head of the updated list.

note: head pointer may be null

Input format

first line N, number of nodes
N subsequent lines contain ~~Integ~~ integers describing the data value of a node being inserted at the list's tail.

Constraints - data elements of linked List argument will always be in non-decreasing order.

Output format: Your remove duplicates function should return the head of the updated linked list

```
public static void main (String args[]) {
    ~~Scan~~
    Scanner sc = new Scanner(System.in);
    Node head = null;
    int T = sc.nextInt();
    While (T-- > 0) {
        int ele = sc.nextInt();
        head = insert (head, ele);
    }
    head = removeDuplicates (head);
    display (head);
}
```

Day 25: Running time and Complexity
2018-01-20

Compile: no code
Run: no code

```
import java.io.*;
import java.util.*;
import java.math.*;
import java.text.*;
import java.util.regex.*;
Public class Solution {
    Static boolean checkPrime (int n){
        if(n==1){
            return false;
        }
        if(n==2){
            return true;
        }
        if(n%2==0){
            return false;
        }
        for(int i=3; i*i<=n ; i+=2){
            if(n%i==0){
                return false
            }
        }
        return true;
    }
    Public static void main(String[] args){
        Scanner SC= new Scanner(System.in);
        int n = SC.nextInt();
        for(int i=0; i<n; i++){
            int val= SC.nextInt();
            if(checkPrime(val)){
                System.out.println("prime");
            }
            else{
                System.out.println("Not prime");
            }
        }
    }
}
```

Task:
A prime number is a natural number greater than 1 that has no positive divisors other than 1 and itself. Given a number, n, determine and print whether it's prime or Not prime

Note: Come up with a $O(\sqrt{n})$ primality algorithm.

Input Format: first line contains integer T, the number of test cases.
Each of the T subsequent lines contains an integer, n, to be tested for primality

Constraints:
- $1 \leq T \leq 30$
- $1 \leq n \leq 2 \times 10^9$

Output Format:
For each test case print whether n is prime or not prime.

2018-01-17 Day 26
: Nested Logic

```
import java.util.*;
import java.io.*;
import java.text.math.*;
import java.math.*
Im...t java.util.regex.*;
Public class Solution {
    Public Static Void main (String[] args) {
        Scanner scan = new Scannar (System.in);
        int rd = scan.nextInt();
        int rm = gcan.nextInt();
        int ry = Scaen.nextInt();
        int ed = Scan.nextInt();
        int em = Scan.nextInt();
        int ey = Scan.nextInt();
        int fine=0;
        Scan.close();
        if(rd > ed && rm == em && ry == ey){
            fine = 15*(rd-ed);
            System.out.println(fine);
        }
        else if (rm > em && ry == ey){
            fine = 500*(rm-em);
            System.out.println(fine);
        }
        else{
            System.out.println(0);
        }
    }
}
```

Compile error: : Scan.next

Submit code: Case 0

31 8 2004
30 1 2004

~~[illegible]~~

- Missing a multiplier for month case

Task:

~~Task~~: Local library
returns fees

1. book returned before
   date fine = 0
2. returned after expected day
   but correct month and year
   fine = 15 × late days
3. after expected month but correct
   year
   fine = 500 × late days
4. after expected year
   fine = 10,000

Input format:

3 space seperated integers denoting
day, month, year, returned
then another 3 for expected return

Constraints

$1 \leq D \leq 31$
$1 \leq M \leq 12$
$1 \leq Y \leq 3000$
guarenteed dates will be
Gregorian Calendar dates

~~Output~~ Output

Print a single integer of return cost.