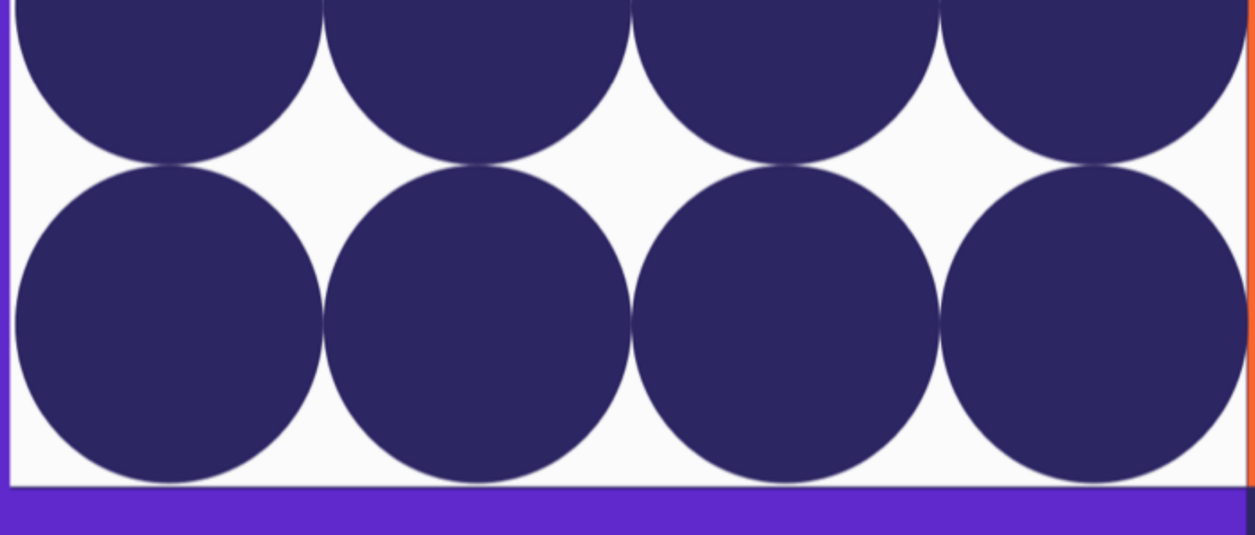
NO. 01

Kunal Jatin Tamhane

Title:

Java Programing

Enjoy the most efficient handwriting experience! Taking notes on the go, whether for inspiration, ideas, knowledge learning, business insights, or even sketches...



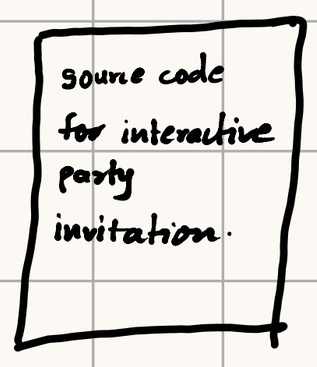
NOTE

Creative notes

By reading we enrich the mind;
by writing we polish it.

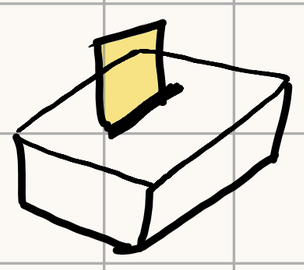
The way Java works:

goal: To write one appliation (in this example an interactive party invitation) and have it work on whatever device your friends has



1. SOURCE

Create a source document. Use an established protocol In our case Java

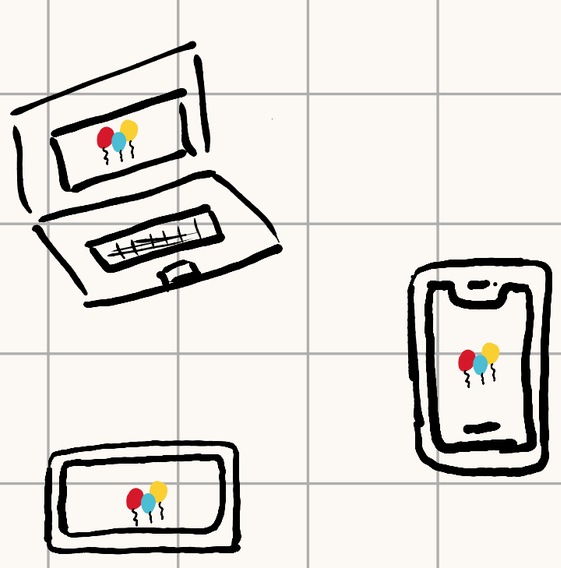


2. COMPILER

Run your source code through compiler The compiler checks for errors and won't let you compile until it is satisfied that everything runs

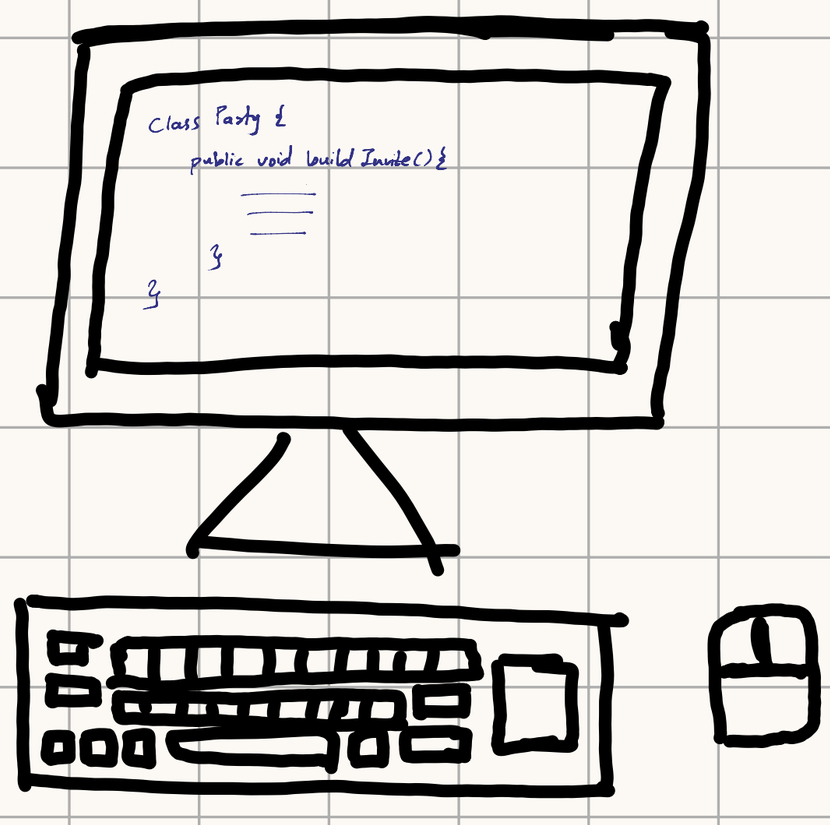
3. OUTPUT

Compiler creates a new document, coded into Java bytecode Any device capable of running Java will be able to interpret (translate) into something it can run The compiled bytecode is platform independent.



4. VIRTUAL MACHINES

Your friends all have java virtual machines (JVM), implemented in software, running inside their electronic gadgets When your friends run your program, the virtual machine reads & runs the bytecode



① Source:
You type source code
Save source code as:
Party.java

② compiler
will compile Party.java file by running javac
If you don't have any errors, You will get a second document named Party.class.

This Party.class file is made up of bytecodes

③ Output (code)
Compiled code without any errors that can run on any machine having JVM.
Party.class

④ Virtual Machines
Run the program by starting the Java Virtual Machine (JVM) with Party.class file The JVM translates bytecode into something the underlying platform understands, and runs your program.

Look how easy it is to write Java

variables

```
int size = 27;
String name = "Fido";
Dog myDog = new Dog(name, size);
x = size - 5;
if (x < 15) myDog.bark(8);


while (x > 3) {
  myDog.play();
}


int[] numList = {2, 4, 6, 8};
System.out.print("Hello");
System.out.print("Dog: " + name);
String num = "8";
int z = Integer.parseInt(num);


try {
  readTheFile("myFile.txt");
}
catch (FileNotFoundException ex) {
  System.out.print("File not found.");
}
```

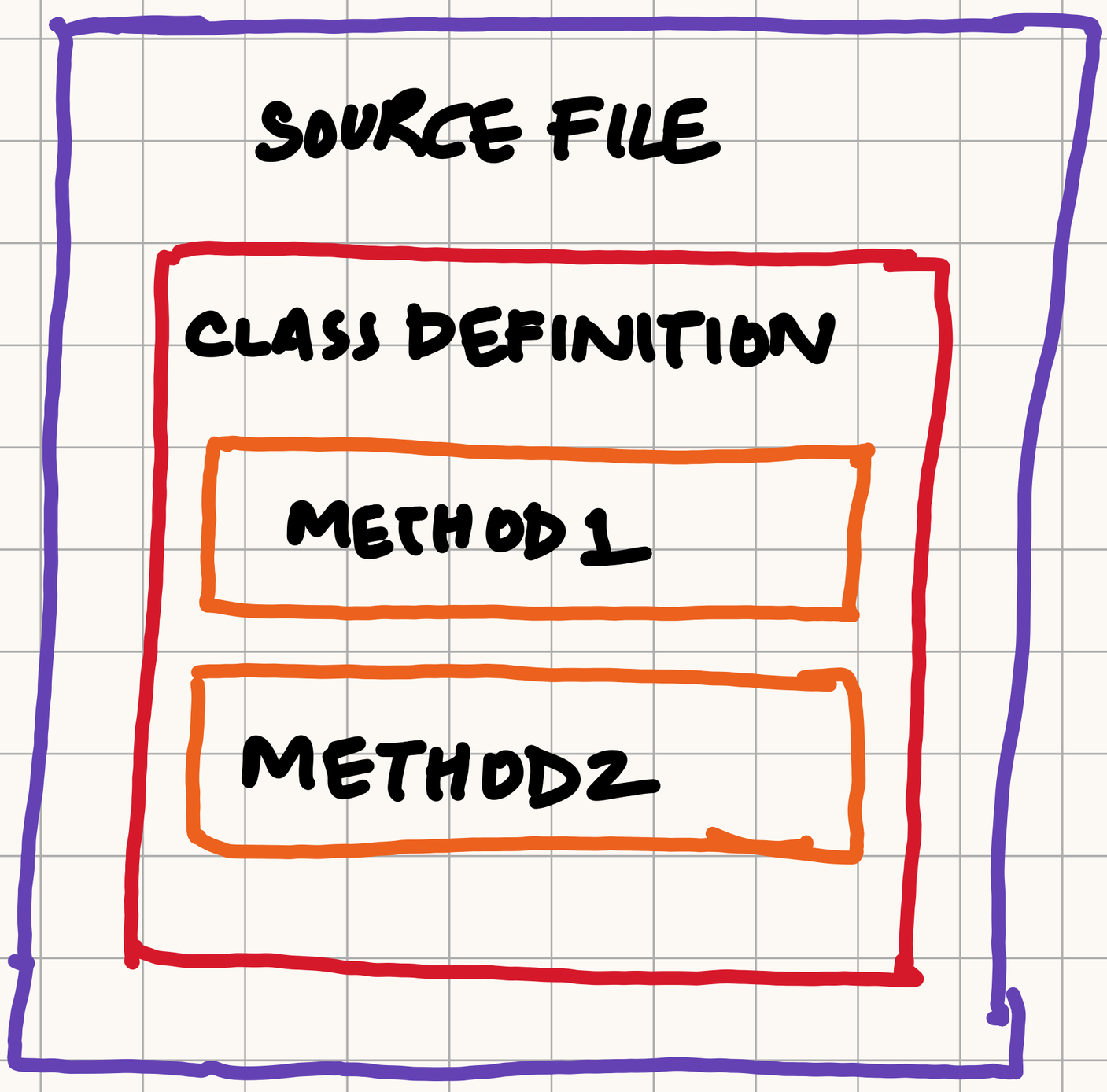
Sharpen your pencil

→ Answers on page 6.

Try to guess what each line of code is doing...
(answers are on the next page).

declare an integer variable named 'size' and give it the value 27
declare a string named 'name' and give it value "Fido"
declare a new dog variable 'myDog' and make new dog using 'name' & 'size'.
subtract size (27) by 5 and store it as x.
if x (value of 22) is less than 15, tell the dog to bark 8 times
keep looping as long as x is greater than 3.
tell dog to play
loop ends. { } loop.
declare list of integers variable 'numList'
print out "Hello"... probably at the command line
print out "Dog: Fido" at command line
declare string character named 'num' give [char] "8".
convert string of characters '8' into actual int value
try to do something
read text file name myFile.txt (try).
end of try.
if the thing you tried didn't worked
If failed print "File not found".

CODE STRUCTURE FOR JAVA



In a sourcefile, put a class

In a class, put method

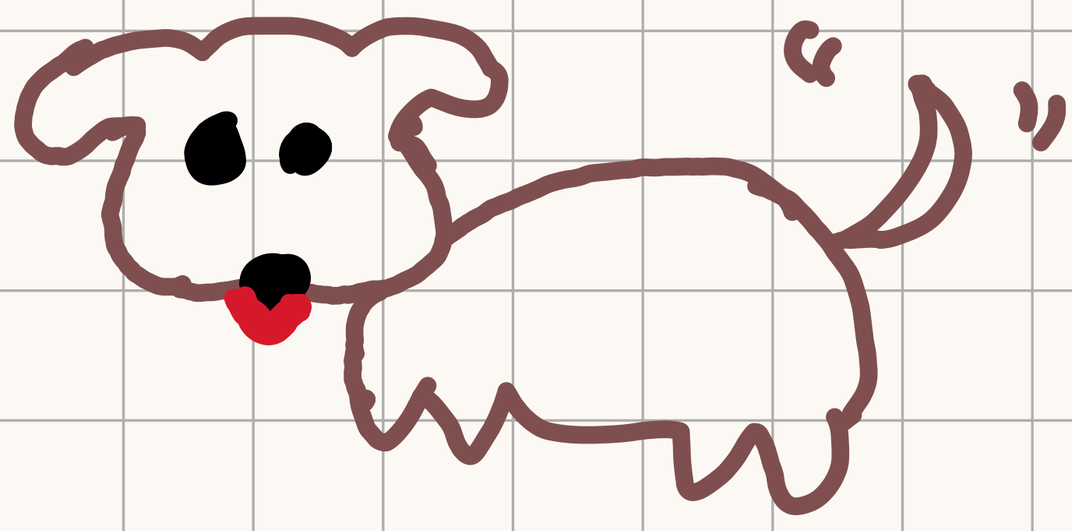
In a method, put statements

What goes inside what with code.

sourcefile:

- A source code file (with .java extension) typically holds one class definition. The class represents a piece of your program.
- The class must start and end with curly braces

```
public class Dog {
```



```
}
```

Class

Max & Elliot.

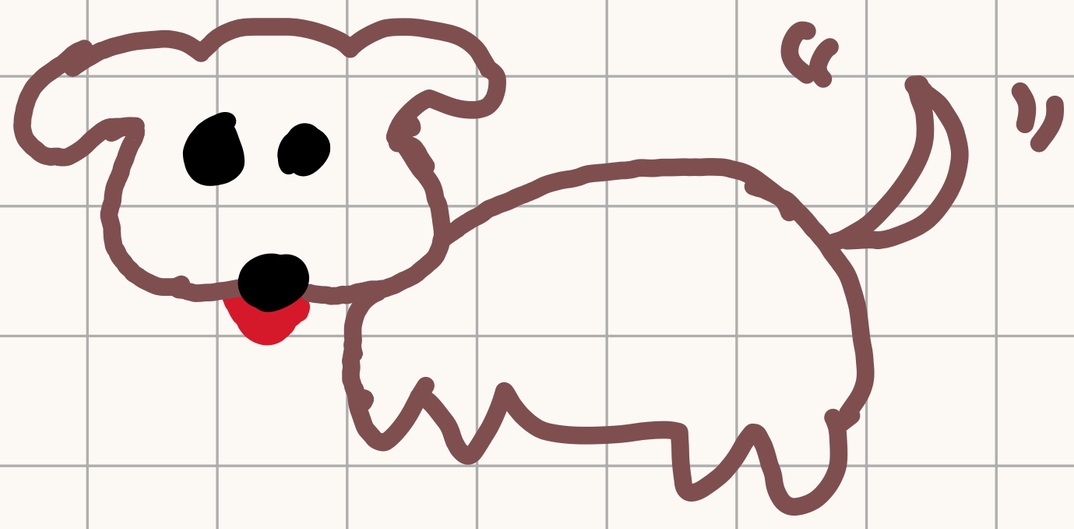
Class.

- A class has one or more methods
- In the Dog class. the bark method will hold instructions for how dog should bark
- Your method must be clearly inside a class, in other words curly braces of class

```
public class Dog {

    void bark() {
        bark!
        bark!
    }

}
```



method

Methods.

- Within curly braces of a method. write your instruction for how that method should be performed
- Method code is basically a set of statements. and for now you can think of a method kind of like a function procedure.

```
public class Dog {
    void bark() {
        statement 1;
        statement 2;
    }
}
```

statements

Anatomy of a Class

Public so everyone can access it

This is a class (duh)

The name of this class

Opening curly brace of the class

```
public class MyFirstApp {
```

(We'll cover this one later.)

The return type. void means there's no return value.

The name of this method

Arguments to the method. This method must be given an array of Strings, and the array will be called 'args'

Opening brace of the method

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

```
        System.out.print("I Rule!");
```

Every statement MUST end in a semicolon!!

This says print to standard output (defaults to command line)

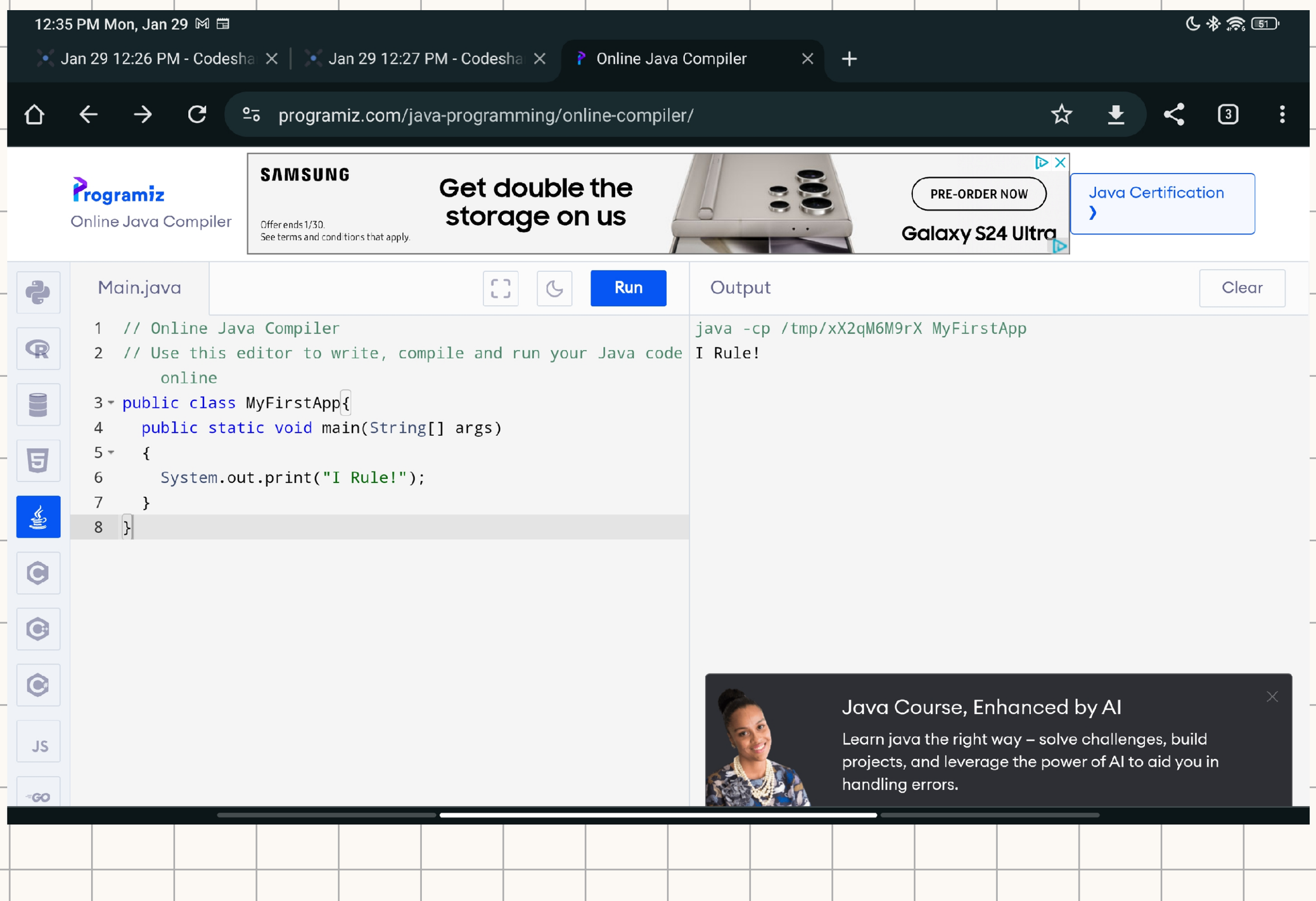
The String you want to print

```
    }
```

Closing brace of the main method

```
}
```

Closing brace of the MyFirstApp class



public so everyone can access it.

Class declaration

braces of class

braces of method

```
public class MyFirstApp {

    public static void main (String [] args) {

        System.out.print ("I Rule!"),

    }
}
```

?? Return type void means there's no return value.

name of this method.

This says print to standard output (defaults to command line).

String which needs to be printed.

- When JVM starts running, it looks for class.

- Then the JVM starts looking for a specifically written method that looks like.

```
public static void main (String[] args)
    {
        // your code goes here.
    }
```

main method

- JVM runs everything between the curly braces {} of your main method

- Every Java application has to have at least one class and atleast one main method

- Once we are inside main, the fun begins, you can say all the normal things that you say in most programming language to make computer do something

- Your code can tell JVM to

a) do something

```
int x = 3;
String name = "Dick";
x = x * 17;
System.out.print("x is " + x);
double d = Math.random();
```

b) do something again and again

```
while (x > 12) {
    x = x - 1;
}
for(int i = 0; i < 10; i = i + 1) {
    System.out.print("i is now" + 1);
}
```

c) do something under this condition

```
if (x == 10) {
    System out print ("x must be 10");
} else {
        System out print ("x isn't 10");
    }
    if ((x<3) && (name.equal ("Dirk")))
        {
            System.out.println ("Gently");
        }
    System out println ("This line runs
        no matter what"),
```

- Facts about Java !!

1) Each statement must end with a semi-colon
2) A single line comment begins with two forward slashes // This is a comment.
3) Most whitespaces doesn't matter x = 3;
4) Variables are declared with a name and type.
5) Classes and methods must be defined within a pair of curly braces

→ {

} ←

Loops and Looping in Java.

- Java has a lot of looping constructs. while, do-while, and for, being the oldest

- You can do a simple boolean test by checking the value of a variable, using a comparision operator like.
< (less than) > (greater than) == (equality)

Example of a while loop.

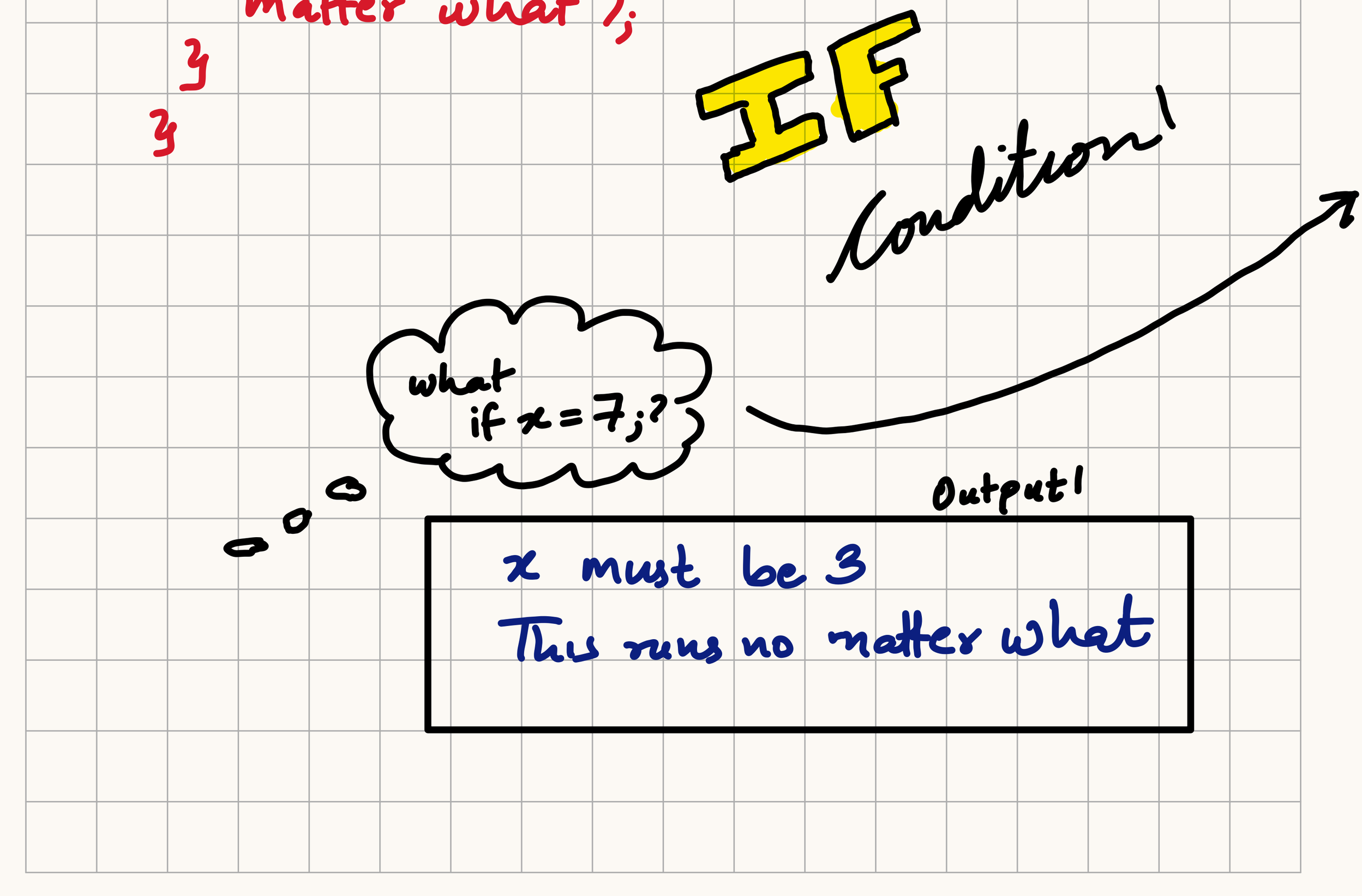
```
public class Loopy {
    public static void main (String[] args) {
      int x = 1,
      System.out println ("Before Loop"),
      while (x<4) {
         System.out println ("In the loop");
       System.out print ln ("value of x is" + x);
       x = x + 1;
    }
      System.out.println ("This is after the loop");
  }
}
```

WHILE loop

Conditional branching

In Java, an if test is basically the same as the boolean test in a while loop - except instead of saying. "while there is still something" you'll say "If there is still "

```
Class IfTest{
    public static void main (String[] args) {
      int x = 3;
      if (x==3) {
          System out println ("x must be 3"),
      }
       System out.println ("This line runs no matter what");
   }
}
```



IF-ELSE Statement!

```
Class IfElse {
    public static void main (string[] args) {
        int x = 2;
        if (x == 3) {
            System out println ("x must be 3");
        } else {
            System.out println ("x is NOT 3");
        }
        System out.println ("This runs no
                             matter what!");
    }
}
```

Output!

```
x is NOT 3
This runs no matter what
```

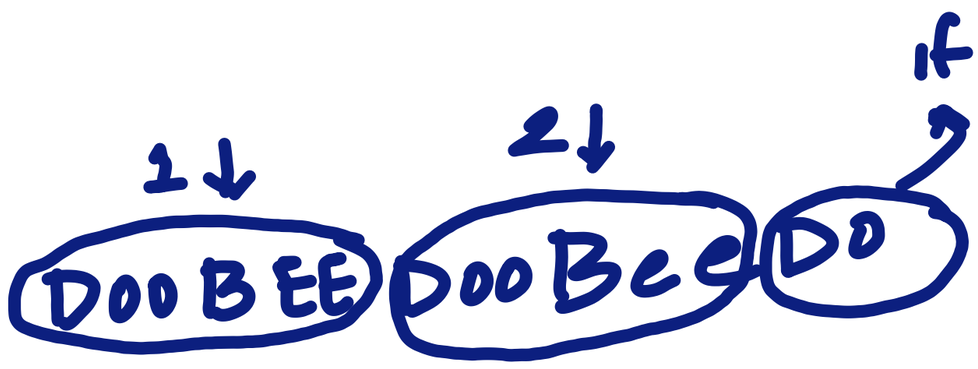
Sharpen your pencil

Given the output:

```
% java DooBee
DooBeeDooBeeDo
```

Fill in the missing code:

```
public class DooBee {
  public static void main(String[] args) {
    int x = 1;
    while (x < ___3___ ) {
      System.out.___print___("Doo");
      System.out.___print___("Bee");
      x = x + 1;
    }
    if (x == ___3___ ) {
      System.out.print("Do");
    }
  }
}
```



Answers on page 25.

If-else condition Statement:

Syntax:

```
if (boolean expression T or F) {
    // body
} else {
    // do this
}
```

Q) Write A program for giving bonus to people above certain salaries

```
public class Main {
        public static void main (String[] args){
    int salary = 25400;
    if (salary > 10000) {
            salary = salary + 2000,
    } else {
            salary = salary + 1000,
    }
    System.out.println (salary);
    }
}
```

Multiple if-else statement.

```
public class Main {
        public static void main (String[] args) {
                int salary = 25400;
                if (salary > 10000){
                salary += 2000;
        } else if (salary > 20000) {
                salary += 3000;
        } else {
                salary += 1000;
        }
    }
}
```

IF-

ELSE-

IF

Loops:

for loop syntax:

```
for (initialisation, condition; increment/decrement)
    {
        // body
    }
```

a) Print numbers from 1 to 5 using for loop

```
public class Main {
    public Static void main (String[] args) {
        for (int num = 1 , num <=5; num+ =1){
            System out println (num);
    }
```

FOR LOOP

Q) Print numbers from 1 to n where n is the input given by the user

```
public class Main {
        public static void main (String [] args){
            Scanner in = new Scanner (System.in);
                int n = in.nextInt();
            for (int num=1; num <=n, num++){
                System.out.print (num + " ");
            }
        }
}
```

INPUT from users

To take input from user you use following Syntax

```
Scanner in = new Scanner (System.in);
    int n = in.nextInt();
```

Specifies input from System.

Can be changed to other data types.

a) Print string "Hello World" n number of times where n is an user defined integer

```
public class Main {
  public static void main (String [] Args) {

        Scanner in = new Scanner (System.in)
            int n = in nextInt();
  for (int num = 1; num <= n; num++) {
            System out println ("Hello World");
    }
  }
}
```

Output

n = 5

Hello World!
Hello World!
Hello World!
Hello World!
Hello World!

While loop Syntax:

```
while (condition) {
    // body
}
```

WHILE loop

a) Print numbers from 1 to 5 using while loop

```
public class Main {
    public static void main (String[] Args) {

        int num = 1;
        while (num <= 5){
            System out println (num);
            num ++;
        }
    }
}
```

DO-WHILE

loop

Syntax

```
do {
    . //body
} while (condition);
```

a) Print numbers from 1 to 5 using do-while loop

```
public class Main {
    public static void main (String [] Args) {

        int n = 1,
        do {
            System out println ( n),
            n = n+ 1;
        } while (n<=5);
    }
}
```

Q) Input 3 numbers from the user and check which one is the largest out of the three

```
public class Main {
    public static void main (String[] args){
        Scanner in = new Scanner (System.in),
        int a = in nextInt (),
        int b = in nextInt (),
        int c = in nextInt ();

 int max = a,
if ( b > max) {
        max = b;
    }
    if (c > max) {
        max = c,
    }
     System out println (max);
  }
}
```

a = 10

b = 20

c = 30

Q) Take character input from the user check whether user's character is upper case or lower case

```
public class Main {
    public static void main (String[] args) {
        Scanner in = new Scanner (System.in);
        char ch = in next() trim() charAt(0);

        if (ch >= 'a' && ch <= 'z') {
            System.out println ("Lower case "),
        }
    else {
            System out println (" Uppercase");
        }
    }
}
```

&& → AND → both conditions needs to be true to make program work

|| → OR → Either one condition needs to be true to make program work

== → equals to

!= → not equal to

Fibonacci Series

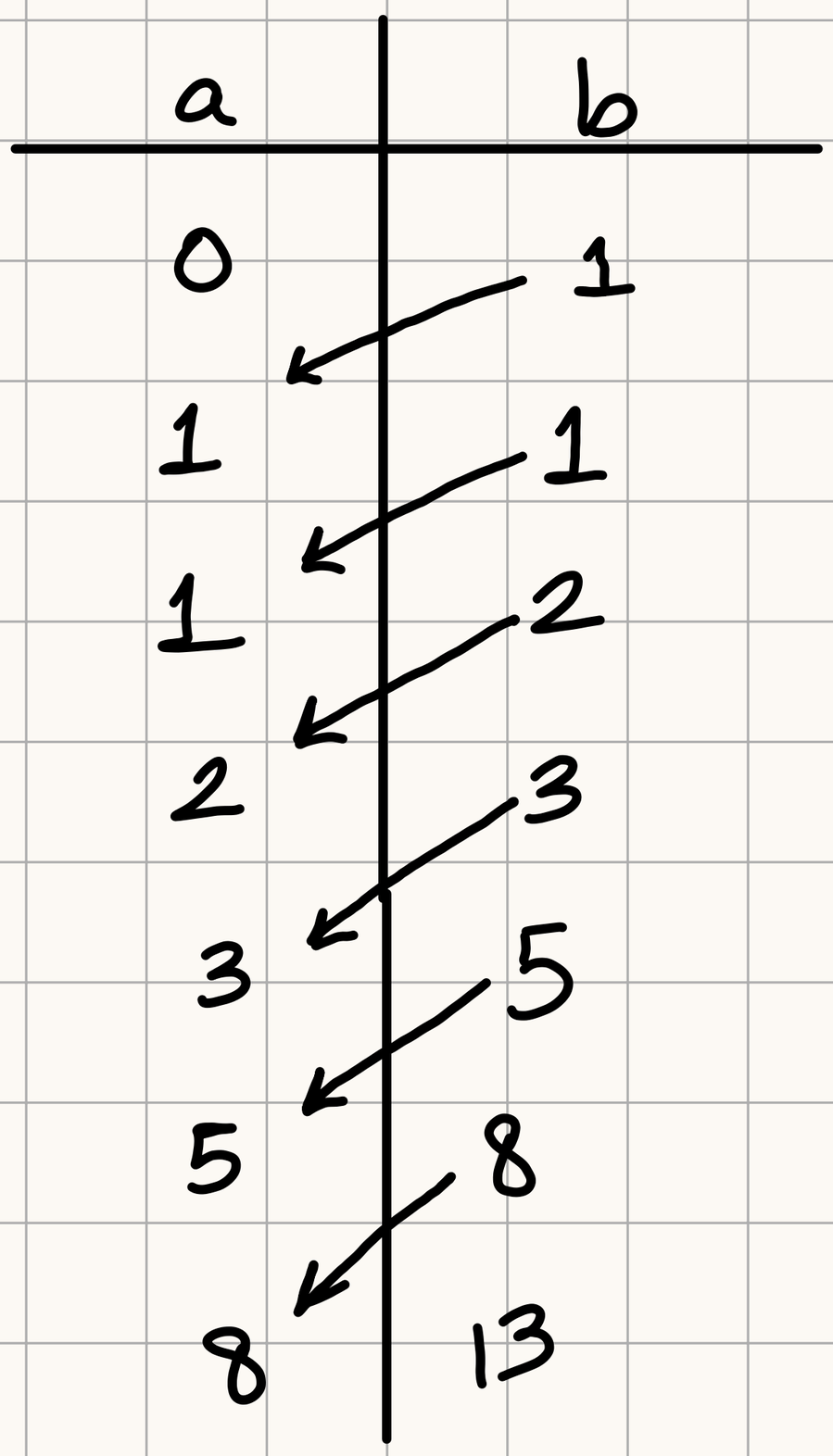
- A sequence of numbers where each number is the sum of the preceding ones

- Starts with 0, 1

0, 1, 1, 2, 3, 5, 8, 13, 21, ... ∞

Code logic

Initate a = 0, b = 1



Q) Write program for Fibonacci number asking nth number from the user

```
import java.util.scanner;

public class Fibo{
    public static void main (String[] args){
        Scanner in = new Scanner (System.in);
         int n  = in.nextInt(),
         int p = 0,    // previous index
        int i = 1;    // current index
        int count = 2,

        while (count <= n) {
            int temp = i;
             i = i + p,
            p = temp,
             count++,
    }
```

a) Take a integer input from the user and reverse the integer to display it.

n = 2 3 5 9 7

n = n/10 → 7

ans = ~~0~~ 7 9 5 3 2

ans = 7 * 10 + 9

= 79

= 79 * 10 + 5

= 795

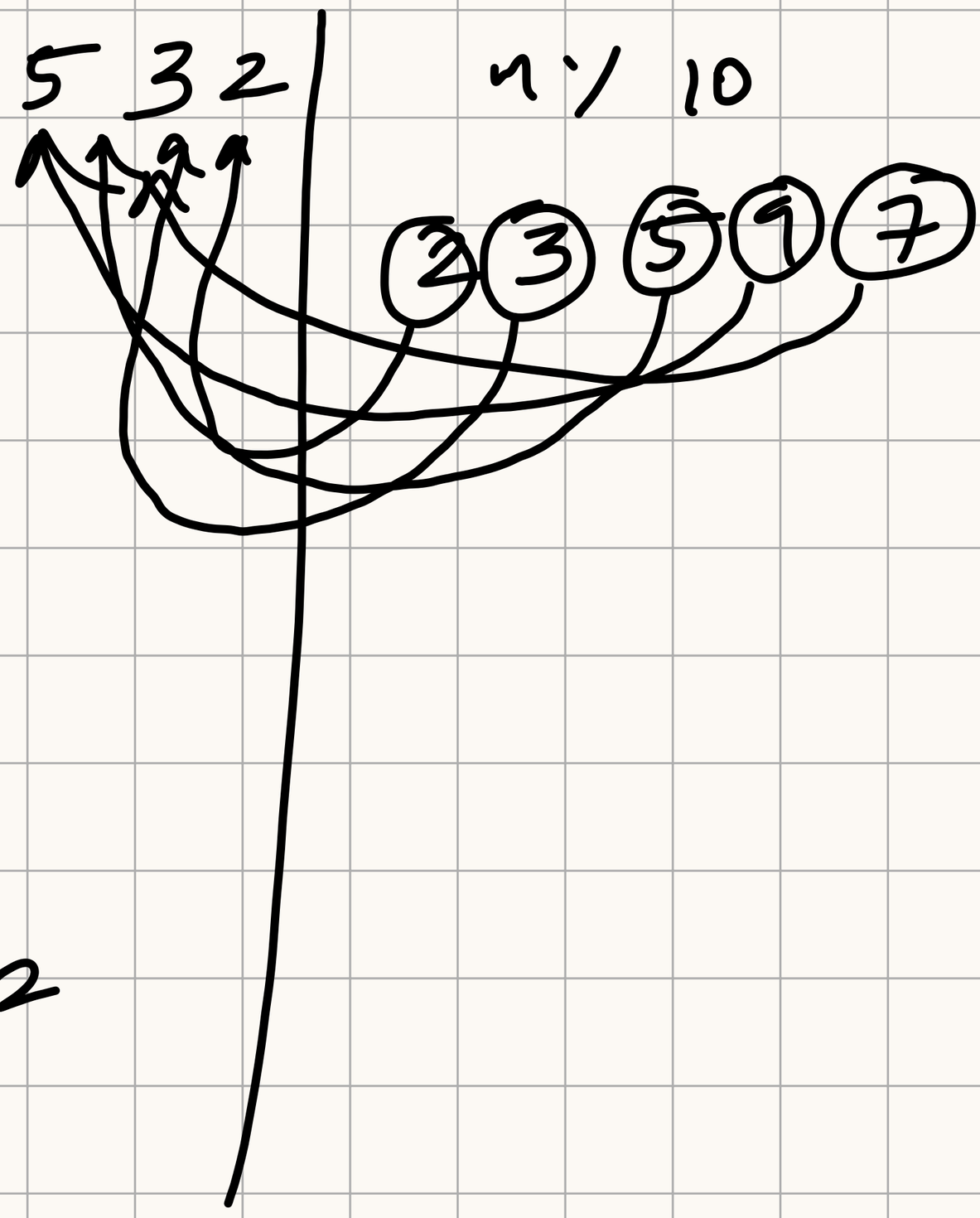
= 795 * 10 + 3

= 7953

= 7953 * 10 + 2

= 79532

Ans. 7 9 5 3 2



logic behind code

```
public class Reverse {
    public static void main (String[] args){
        int num = 28479;
        int ans = 0;

        while (num >0) {
            int rem = num % 10;
            num /=10;

            ans= ans * 10 + rem;
        }
        System out println (ans),
    }
}
```

Calculator program

```
public class Calculator {
    public static void main (String[] Args) {
        Scanner in = new Scanner(System.in),
        int ans = 0,
        while (true) { System.out.println ("Enter op");
            char op = in.next().trim().charAt(0),
            if (op=='+' || op=='-' || op=='*' ||
                op=='/' || op=='%') {

            System.out.println("Enter 2 numbers"),
                int num1 = in.nextInt();
            int num2 = in.nextInt(),
                if (op=='+') {
                    ans = num1+num2;
                }
            if (op=='-') {
                    ans = num1-num2;
                }
            if (op=='*') {
                    ans = num1*num2;
                }
            if (op=='/') {
                if (num2!=0) {
                    ans = num1/num2,
            }
        }
```

```
            if (op == '/') {
                ans = num1 /. num2;
            }
        }
    } else if (op == 'x' || op == 'X') {
        break;
    }
    else { System.out.println ("Invalid input"); }
    System.out.println (ans);
  }
}
```

* Switch statement

Syntax

```
switch (expression){
    // cases
 case one
       // do something,
       break,
 case two
     // do something;
      break;
default :
 // do something }
```

a) Write a program in which the user inputs a fruit and the program returns characteristics of the fruit For example · Input = Mango, Output = King of fruits, Input = Apple Output = sweet red fruit

```
public class Main {
    public static void main (String[] args) {

        Scanner in = new Scanner (System.in);
        String fruit = in.next();

        switch (fruit){
            case "Mango"
                System.out.println ("King of fruits");
                break;

            Case "Apple"
                System.out.println ("Red sweet");
                break;

            Case "Orange".
                System.out.println ("Sweet, sour, round");
                break;

            Case "Grape".
                System.out.println ("small, round, many");
                break;
            default.
                System.out.println ("Enter valid fruit");
        }
    }
}
```

* Nested Switch Case.

```
public class Main {
  public static void main (String[] args) {

      Scanner in = new Scanner (System.in);
      int empID = in.nextInt ();
      String department = in.next ();

      switch (empID) {
          case 1:
              System.out.println ("Kunal Tamhane");
              break;
          case 2
              System.out.println (" Rohit Maity ");
               break;
          case 3
              switch (department){
                  case "IT"
                      System.out.println ("IT Department");
                       break;
                  case "Management":
                       System.out.println ("Management Dept");
                       break;
                  default
                       System.out.println ("Enter a dept")
              }
              break;
          default  —"—  }}}
```

Functions or Methods in Java!

- A method is a block of code which only runs when it is called

- You can pass data, known as parameters, into a method

- Methods are used to perform certain actions, and they are also known as functions

- A method must be declared within a class It is defined with the name of method, followed by ()

- Creating a method.

```
public class Main {
  static void myMethod() {
    // code to be executed
  }
}
```

```
return_type  name (arguments){
        // body
        return statement;
  }
```

- myMethod() · is the name of the method
- static means that method belongs to the Main class and not an object of main class
- void means that this method doesnot returns any value

★ Calling a Method

To call a method in java, write the method's name followed by two parantheses () and a semi-colon,

Example·

```
public class Main {
    static void myMethod() {
        System.out.println("I just got executed!");
    }

    public static void main(String[] args) {
        myMethod();
    }
}

// Outputs "I just got executed!"
```

- A method can be called multiple times.

Example

```
public class Main {
    static void myMethod() {
        System.out.println("I just got executed!");
    }

    public static void main(String[] args) {
        myMethod();
        myMethod();
        myMethod();
    }
}
```

Output:

I just got executed
I just got executed
I just got executed

a) Write a program creating a function (method) named greeting When greeting() is called the program should output "Hello World!"

```
public class Greeting {
        public static void main (String [] args) {
            greeting();
        }
    static void greeting() {
            System.out.println ("Hello World!");
        }
}
```

★ Returning values in Methods.

- The **return** keyword finishes the execution of a method and can be used to return a value from a method

- The **void** keyword is used to specify that a method shouldnot have a return value

- Return is used to get the value from the program and store it.

Example.

a) Write a function to perform sum of two number store the answer of sum in return value of method

```
public class sum {

    public static void main (String [] args) {
        int ans = sum2();
        System.out.println (ans);
    }

    ← return datatype.
Static int sum2 () {
        Scanner in = new Scanner (System in);
        System out println ( " Enter number 1 ");
        int num1 = in nextInt ();
        System out print ( " Enter number 2. ");
        int num2 = in nextInt ();
        int sum = num1 + num2;
        return sum;
       // after return nothing can be executed in method

    }

}
```

Example:

Q) Write a program to return string by a method

```
public class StringExample {
    public static void main (String[] args) {

        String message = greet();
        System.out.println(message);

    }

    static String greet() {
        String greeting = "how are you";
        return greeting;
    }
}
```

```
public class Sum {
    public static void main (String [] args){

        int ans = sum3 (20, 30),
        System.out.println (ans);
    }

    static int Sum3 (int a, int b) {
        int sum = int a + int b;
        return sum,
    }
```

```
public class StringExample {
    public static void main (String[] Args){

        String personalised = myGreet ("Kunal"),
    }

    Static String myGreet (String name){

        Stnng message = "Hello" + name,
        return message;
    }
```

Q) Make a personlised greeting using methods In this method the program should greet the user with his name For example.
Input "Kunal" Output "Hello Kunal!"

```
public class StringExample {
    public static void main (String[] args) {

        Scanner in = new Scanner (System.in)
        System.out.println ("Enter your name: ");
        String name = in.next();
        String personalised = myGreet (name),
         System.out.println (personalised);
}
 static String myGreet (String name) {
        String message = "Hello" + name,
        return message;
    }
```

Q) Swapping two numbers in Java

```
public class Swapping {
    public static void main (String [] args) {
        int a = 10,
        int b = 20,

        int temp = a,
        a = b,
        b = temp;

        System.out.println (a + "  " + b),
    }
}
```

Passing Values in Java

```
main ( ) {
        name = "Kunal",
        greet (name),
    }

greet (naam) {
        print (naam)
    }
```

Object

// when name is passed in this method the value of reference variable is paised

copy of value of refrence variable is passed

stack

Heap

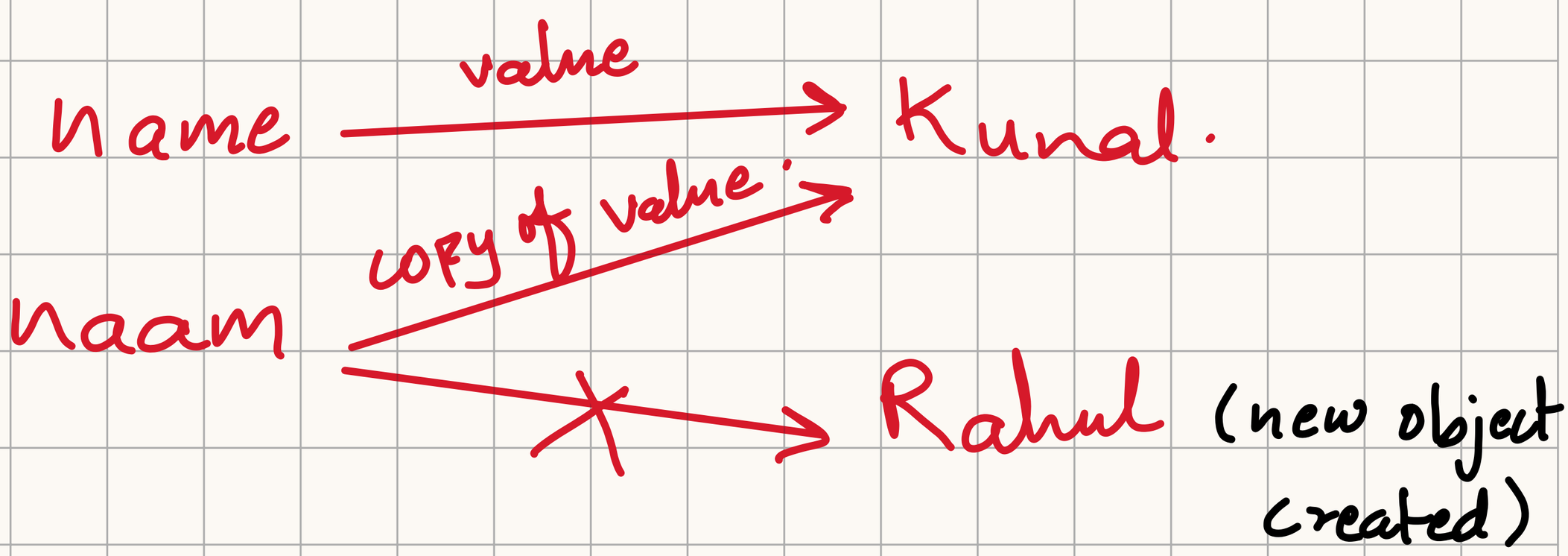
name → value → "Kunal"

naam → copy of value.

- In java there is no pass by reference there is pass by value

```
public static void main ( ) {
        name = Kunal
        change (name),
        print ( name ),  → Kunal (O/P)
}

        change (naam) {
            naam = Rahul
    }
```



QUESTION?

Q) Will it get Swapped ?

```
class main {
    public static void main (String [] Args) {
        int a = 10;
        int b = 20;
        int  swapping = Swap (int a, int b);
        system out println (swapping)
```

```
Static void swap (int a, int b) {
    int temp = a;
      a = b;
    b = temp;
}
```

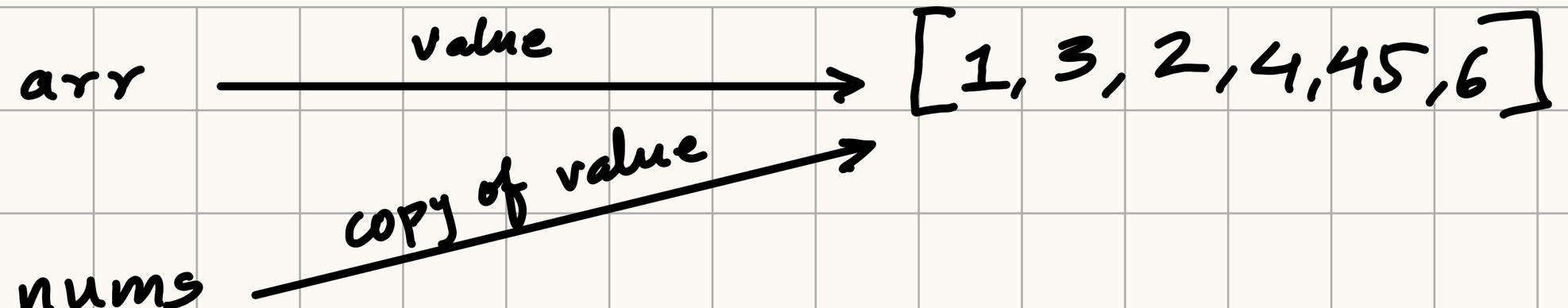
Answer No it won't Swap

- for primitives (int, short, char, bool etc) is pass by value (passing actual value)
- for objects and strings Java passes value of reference variable like (name, naam)

* Lets look into an example where original value is changed

```
public class Main {
    public static void main (String [] args) {
        int [] arr = {1,3,2,45,6 };
        change (arr);
        System out println (Arrays toString (arr));
    }

    static void change (int [] nums) {
        nums [0] = 99;
    }
}
```



nums[0] = 99 ⟶ object getting modified

- Here the object is not getting changed Instead the same object is getting modified

- If you make changes to the object via this ref variable, same object will be changed.

Scoping in Java:

In Java, variables are only accessible inside the region they are created This is called scope.

a) Method Scope:

Variables declared directly inside a method are available anywhere in the method following the line of code in which they were declared

```
public class Main {

    public static void main (String [] args) {
        // code here cannot use x

        int x = 100,

      // code here can use x

       System out println (x),
      }
}
```

b) Block Scope:

A block of code refers to all of the code between curly braces { }

Variable declared inside blocks of code are only accessible by the code between the curly braces, which follows the line in which variable was declared

```
public class Main {
    public static void main (Strng [] args) {
        // code here cannot use x
    { // this is a block
        //code here cannot use x
        int x = 100,
        // code here can use x
        System.out.println (x);
    } // The block ends here
    // Code here cannot use x
}
}
```

9) Loop Scope

Scoping in for loop.

```
for (int i=0, i<4; i++){
        System.out.println(i);
}  // i cannot be used outside this
```

Shadowing

Variable Argument (VarArgs)

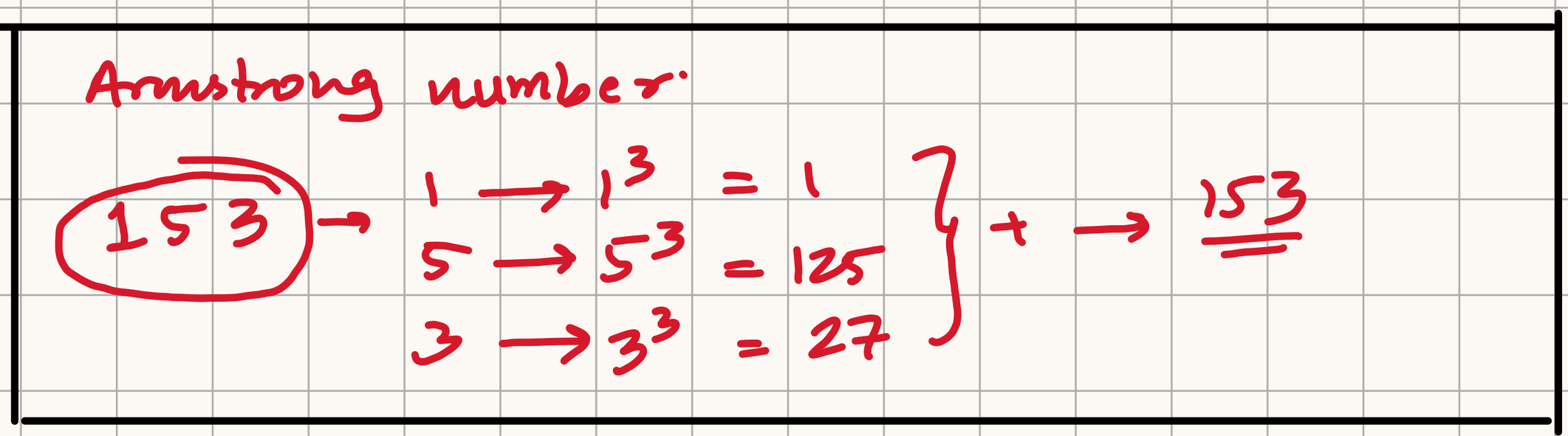
Method overloading

Q) Write a program to determine prime number by using methods in java

```
public class Prime {
    public static void main (String[] args){
        Scanner in = new Scanner (system in);
        int n= in nextInt();
        boolean ans = is Prime (n),
       System out println (isPrime(n)),
    }
    Static boolean isPrime (int n) {
     if (n<=1) {
        return false;
     }
     int c = 2,
    while (c * c <= n) {
        if (n % c == 0) {
            return false.
        }
        c++,
    }

    if (c*c > n) {
        return true,
      }
        return false,
} }
```

Q) Write a Program to check Armstrong Numbers




```
a = 153
while (a > 0) {
        rem = a % 10
        cube = rem
        sum + = cube
        a = a / 10
}
```

```
public class Main {
    public static void main (String[] args) {
        Scanner in = new Scanner (System.in),
        int n= in nextInt(),
        for (int i = 100; i<1000; i++) {
            if (isArmstrong (i)) {
                System out println (i),
    } } }
}
static boolean isArmstrong (int n) {
    int original =n;
    int sum = 0;
    while (n >0) {
        int rem = n % 10,
        n=n /10,
        sum= sum + rem*rem*rem;
    }
    if (sum == original ) {
        return true,
    }
    return false; }
```