Get Smart: With Java Programming

Yaman Omar Alashqar

Computer Science

```
System.out.println("WELCOME TO THIS COURSE\n");
```

MY FIRST JAVA PROGRAM

Cornell Notes

1. Take notes in the largest section of the page.

While listening to a lecture, take notes in the right-hand section of the page.

- Include any information that is important in the slideshow.
- listen out for points that are emphasized or repeated, as these are likely important.

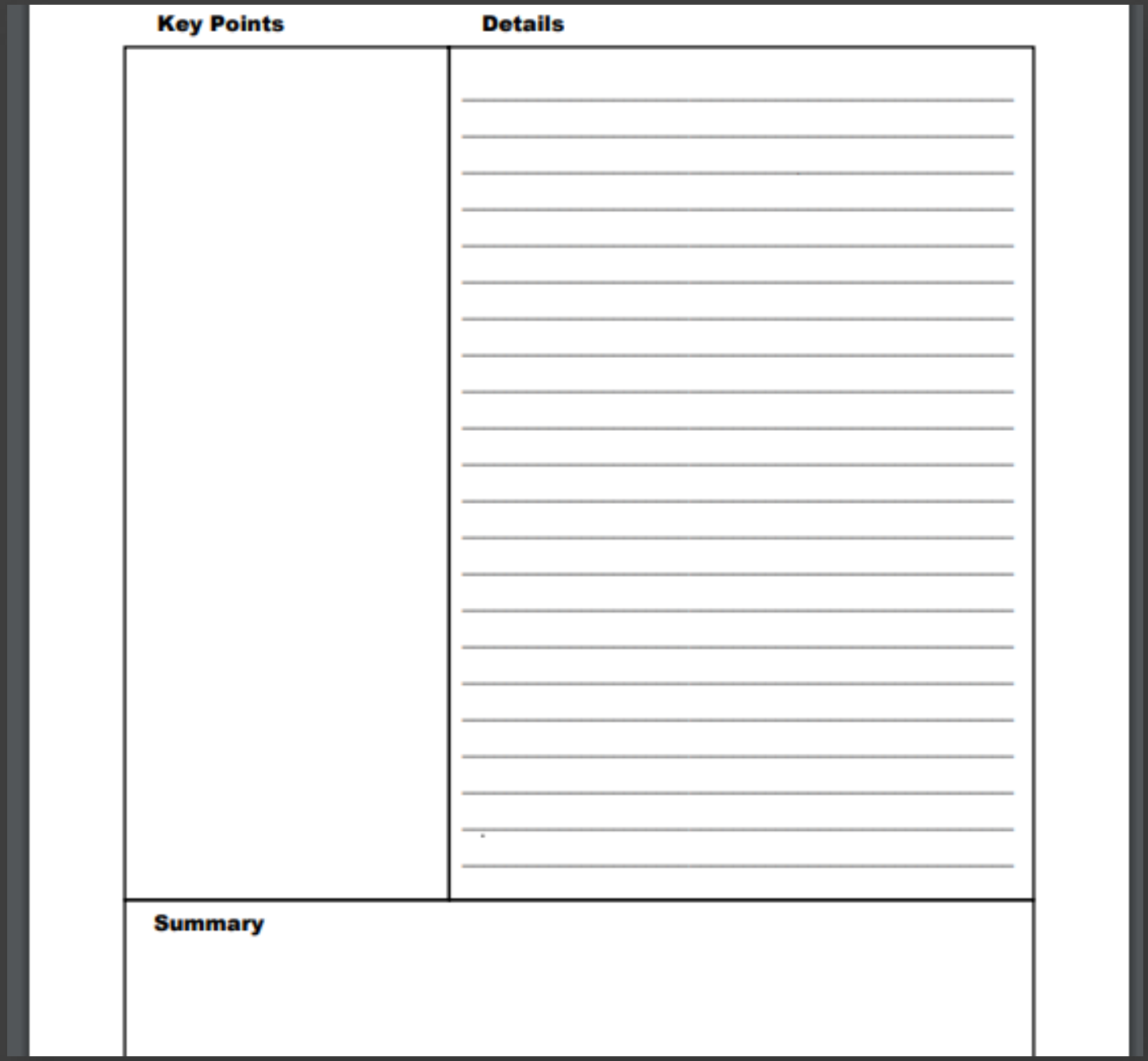
2. Key points & Questions

After the lecture summarize the key points.

- Write potential questions
- Short keywords of the main ideas

3. Summarize the main ideas

Just do it in your own words, imagine you are explaining it to somebody else



What's Tonight's Menu?

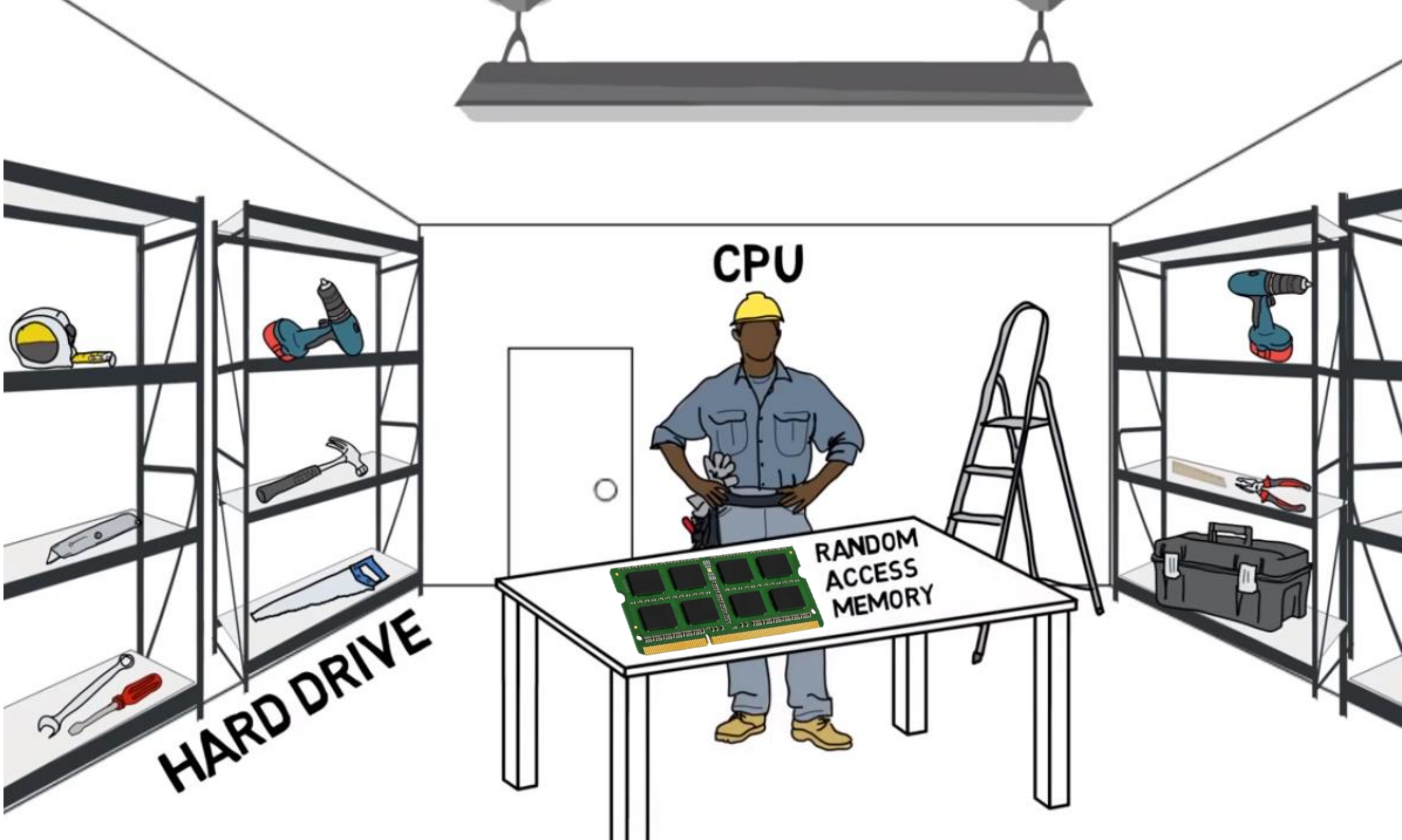
Basic Memory Concepts

Display Output (Console)

Input From The Keyboard

Booleans & Conditions

BACK TO MEMORY CONCEPTS


CPU
RANDOM
ACCESS
MEMORY
HARD DRIVE

The **computer stores data** internally in **memory locations**.
These data are found by the **variable** names used by a program.

Each variable name is given a memory location,
and each memory location can hold one and only one value at a time.

When a new value is assigned to the variable location,
the previous value is destroyed.

These memory locations are temporary, as the internal memory is a volatile memory.

When a program completes its instructions, and/or when the computer is turned off, the values stored in the internal memory are destroyed.

* Data and instructions are temporarily stored in the computer's internal memory during the processing.
- When data, information, or programs have to be kept for future Data Storage use, they are stored on an external storage medium such as a hard disk drive in storage areas called files.

Declaring a variable

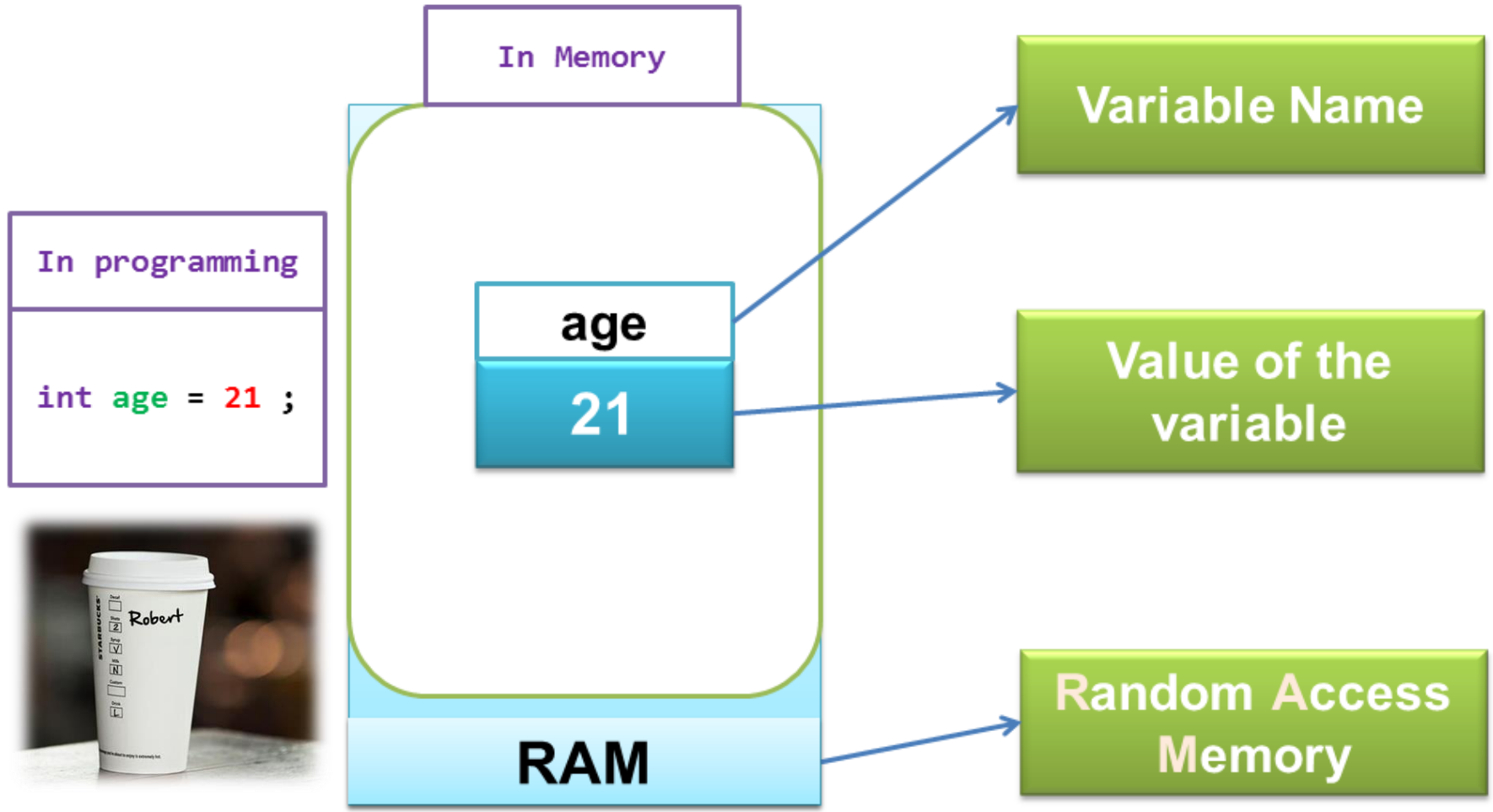
A variable declaration statement typically causes the variable to be created in memory.

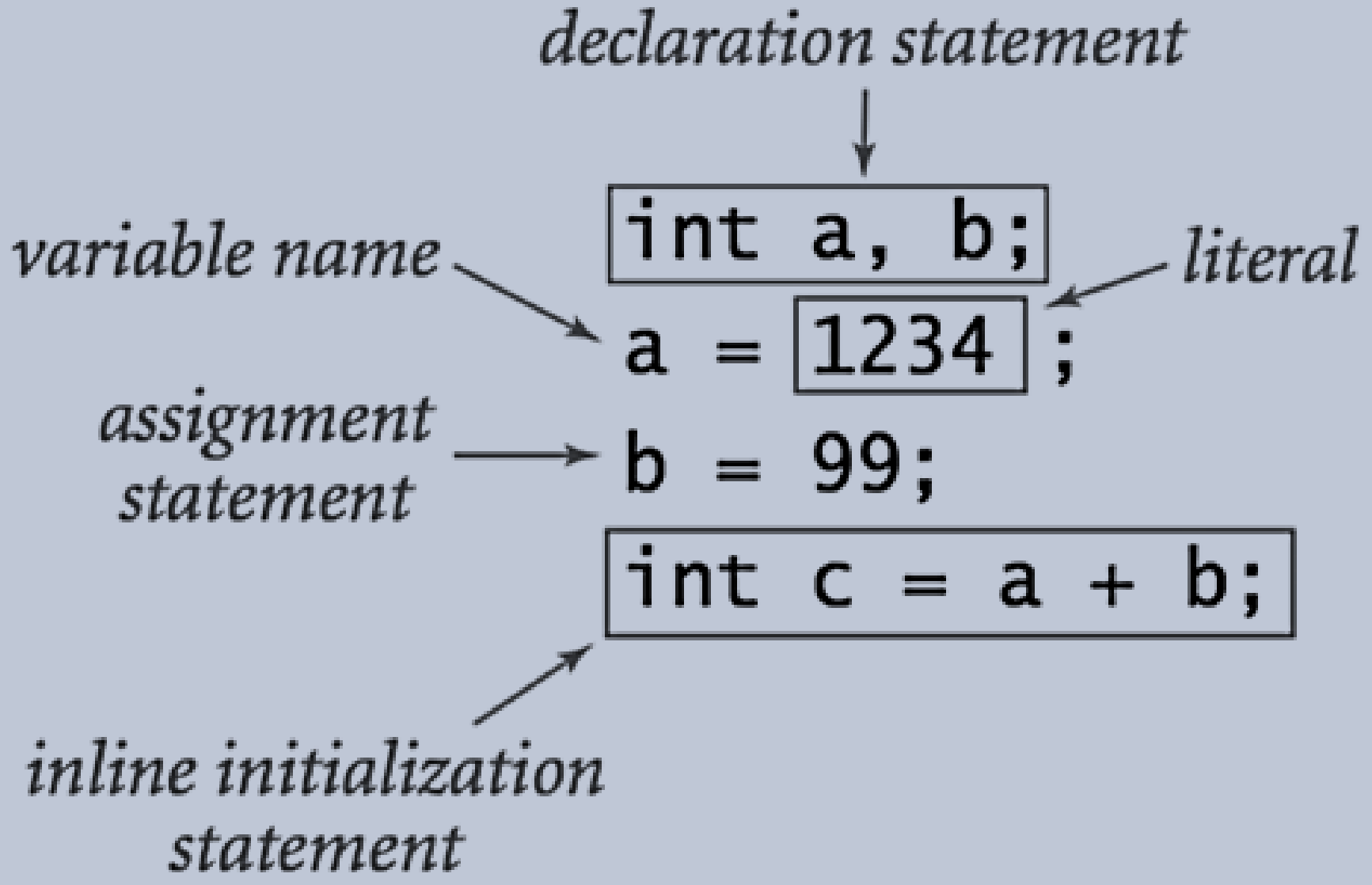
```
data_type var_name;
```

data_type
int
double
char
boolean

Identifiers are the names of variables, methods, classes, packages

- Series of characters (letters , digits , underscores, dollar sign)
- Does not start with a digit
- Case sensitive
- Not a Reserved Java word
- No spaces allowed


In Memory
In programming
int age = 21 ;
age
21
RAM
Variable Name
Value of the variable
Random Access Memory
Robert


declaration statement
int a, b;
variable name
literal
a = 1234;
assignment statement
b = 99;
int c = a + b;
inline initialization statement

1. Declaration: creating a variable as each variable used in a program must be declared.
E.g. int a;
Basically its creating a new variable (declaring) so that it can be used in further parts of the program.

2. Assignment: if we need to put some value in a variable,
E.g. a = 10;
Assignment can be done n number of times throughout the program.

3. Initialization: declaring a variable and assigning a value at the declaration time
E.g. int a=10;.
Initialization = Declaration + Assignment in a single step.

The **computer stores data** internally in **memory locations**.
These data are found by the **variable** names used by a program.

Each variable name is given a memory location,
and each memory location can hold one and only one value at a time.

When a user enters a new value into the variable location, the previous value is destroyed.

These memory locations are temporary, as the internal memory is a volatile memory.
When a program completes its instructions, and/or when the computer is turned off, the values stored in the internal memory are destroyed.

* Data and instructions are temporarily stored in the computer's internal memory during the processing.
* When data, information, or programs have to be kept for future Data Storage use, they are stored externally on an external storage medium such as a hard disk drive in storage areas called files.
* There are basically two types of files: program files and data files.
* Program files contain the instructions to tell the computer what to do.

IN CLASS TASK:

Think of any real world application or program, and define all the possible variables that we might need.

Select all statements where the declaration type does NOT match the value !

- [ ] `char me = 'I';`
- [ ] `boolean fact = true;`
- [ ] `boolean number = 17;`
- [ ] `String text = "text";`
- [ ] `double price = 23.75;`
- [ ] `long total = 100.1;`

What do you think this program will print?

```
int x = 0;
int y = 4;
double z = 3;
x = x + 2;
z = x + y - 7;
y = x * 3;
System.out.println("x = "+x);
System.out.println("y = "+y);
System.out.println("z = "+z);
```

- ○
```
x = 2
y = 0
z = -3.0
```
- ○
```
x = 0
y = 0
z = 1.5
```
- ○
```
x = 0
y = 4
z = 3.0
```
- ○
```
x = 2
y = 6
z = -1.0
```

ORACLE Java Documentation

The Java™ Tutorials

Search the online Java Tutorials Submit

Hide TOC

The Java Tutorials have been written for JDK 8. Examples and practices described in this page don't take advantage of improvements introduced in later releases and might use technology no longer available.
See JDK Release Notes for information about new features, enhancements, and removed or deprecated options for all JDK releases.

Variables

As you learned in the previous lesson, an object stores its state in *fields*.

```
int cadence = 0;
int speed = 0;
int gear = 1;
```

The What Is an Object? discussion introduced you to fields, but you probably have still a few questions, such as: What are the rules and conventions for naming a field? Besides `int`, what other data types are there? Do fields have to be initialized when they are declared? Are fields assigned a default value if they are not explicitly initialized? We'll explore the answers to such questions in this lesson, but before we do, there are a few technical distinctions you must first become aware of. In the Java programming language, the terms "field" and "variable" are both used; this is a common source of confusion among new developers, since both often seem to refer to the same thing.

The Java programming language defines the following kinds of variables:

- **Instance Variables (Non-Static Fields)** Technically speaking, objects store their individual states in "non-static fields", that is, fields declared without the `static` keyword. Non-static fields are also known as *instance variables* because their values are unique to each *instance* of a class (to each object, in other words); the `currentSpeed` of one bicycle is independent

https://docs.oracle.com/javase/tutorial/java/nutsandbolts/variables.html

ARATHMITIC

Addition	+	$f + 7$	`f + 7`
Subtraction	–	$p - c$	`p - c`
Multiplication	*	bm	`b * m`
Division	/	x / y or $\frac{x}{y}$ or $x \div y$	`x / y`
Remainder	%	$r \bmod s$	`r % s`

Algebra: $z = pr\%q + w/x - y$

Java:

```
z = p * r % q + w / x - y;
```

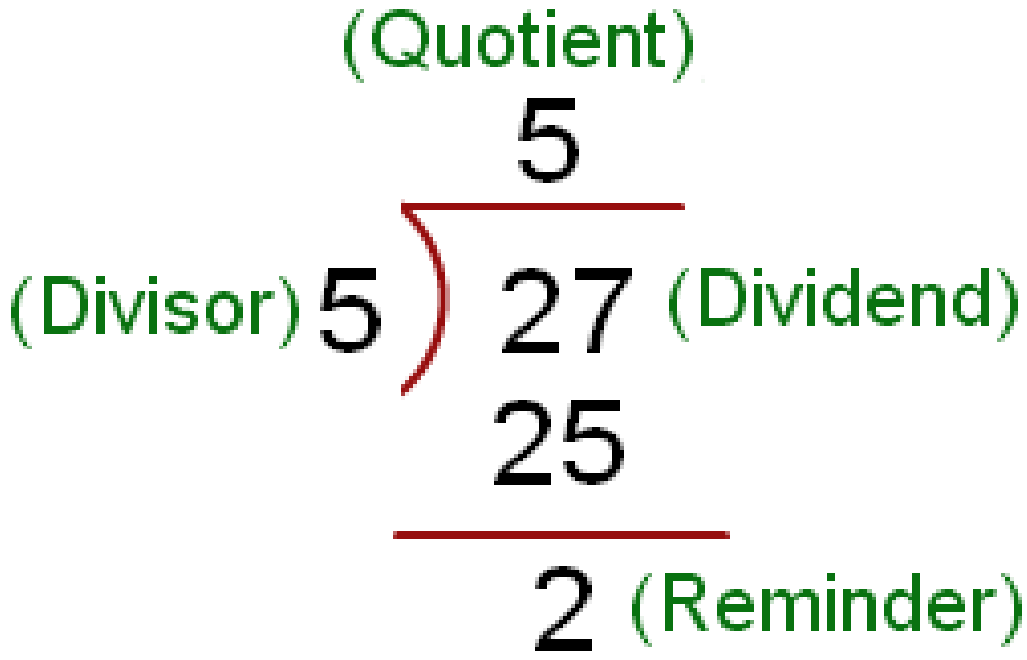
6 1 2 4 3 5

Priority of operation

Parentheses	()
logical NOT	!
Multiplicative	* / %
Additive	+ -
Relational	< > <= >=
Equality	== !=
logical AND	&&
logical OR	||
Assignment	= += -= *= /= %=

Operator	Meaning	Example
+	Addition	4 + 7 ⟶ 11
-	Subtraction	12 - 5 ⟶ 7
*	Multiplication	6 * 6 ⟶ 36
/	Division	30 / 5 ⟶ 6
%	Modulus	10 % 4 ⟶ 2

Java doesn't provide the power operator. (3^4).
You have to use java.lang.Math.pow(3,4).
That method returns type "double"



8/3 = 2
but 8.0/3 = 2.2666

So:
int x = 8; int y=3;
int z = x/y;
=2

double x = 8; int y=3;
double z = x/y;
=2.2666

Displaying Output on the console

```
System.out.println("Hello, World!");
```

- **Java is case sensitive ~~system.out.println();~~**

- **Any characters inside the quotation marks are displayed as it is**
  (these are called strings well discus them later)

- **A semi-colon indicates that it is the end of statement**

Reading Numbers from the Keyboard

```
Scanner input = new Scanner(System.in);

System.out.print("Enter an int value: ");
int intValue = input.nextInt();
```

- nextInt(): reads an integer of the int type.
- nextFloat(): reads a number of the float type.
- nextDouble(): reads a number of the double type

Input/Output

- INPUT
- READ
- SCAN

To get/read values from the user
These values are assigned to a variable

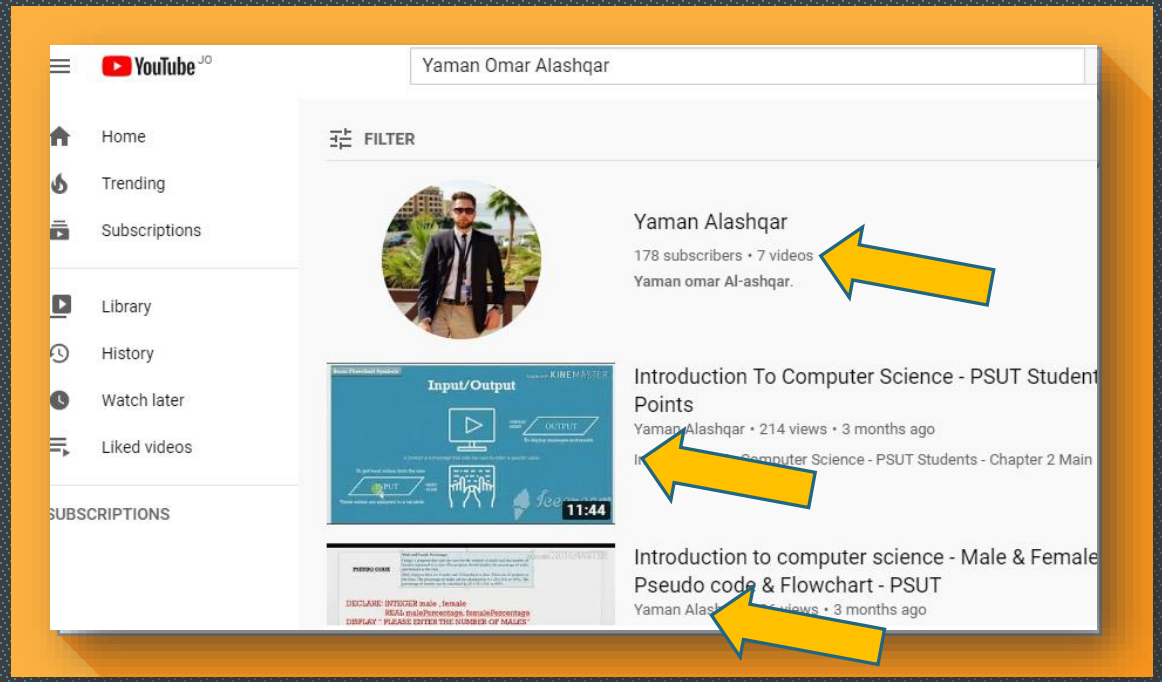
Scanner

- OUTPUT
- DISPLAY
- PRINT

To display massages and results
Display Text + Variables

System.out.println

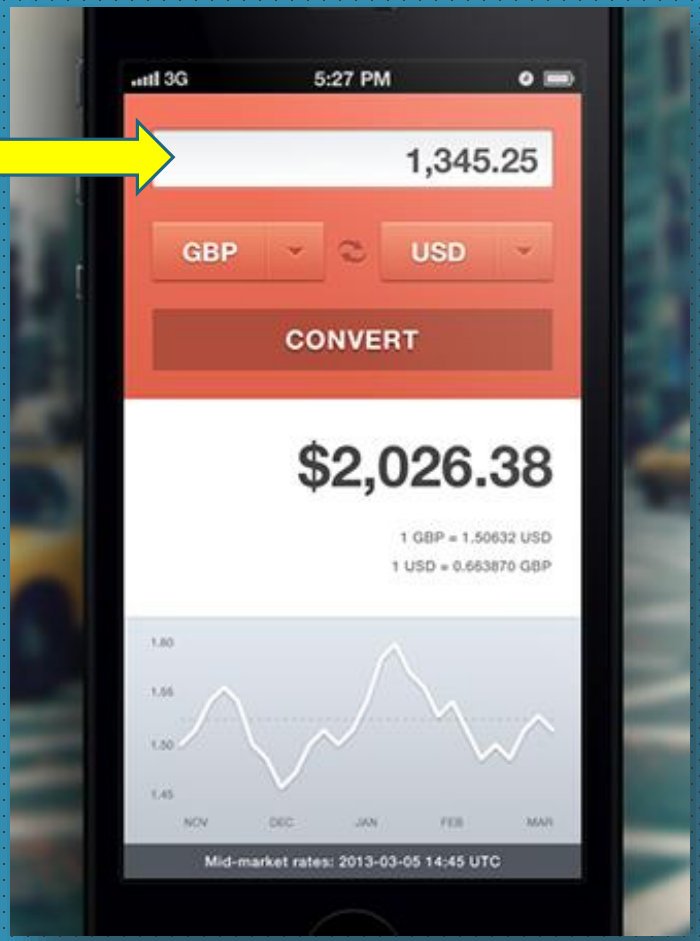
Input/Output

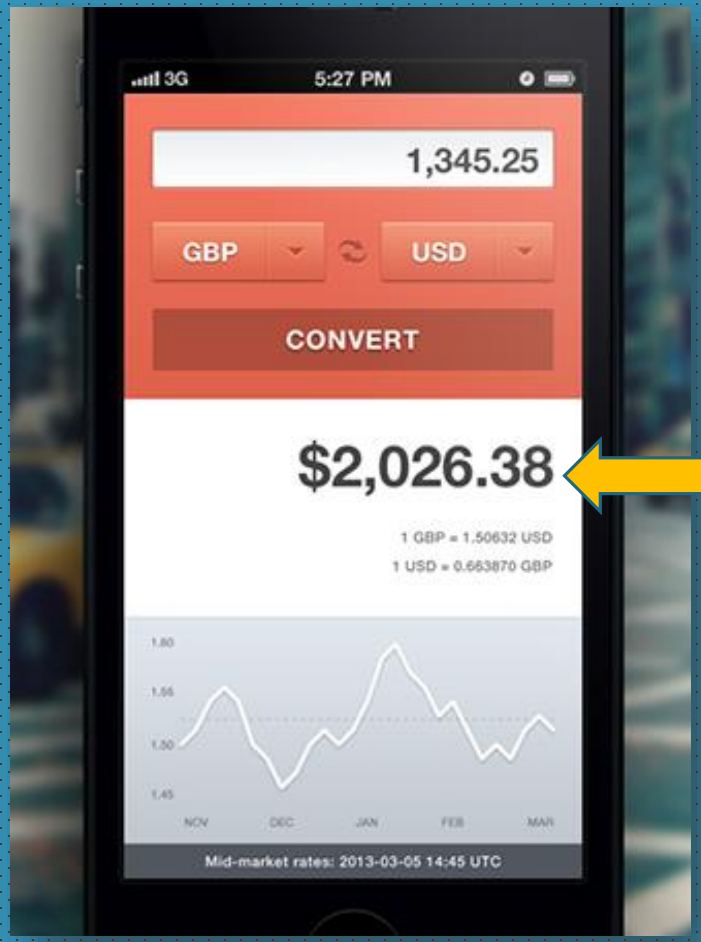


To get/read values from the user

To display massages and results

Input/Output



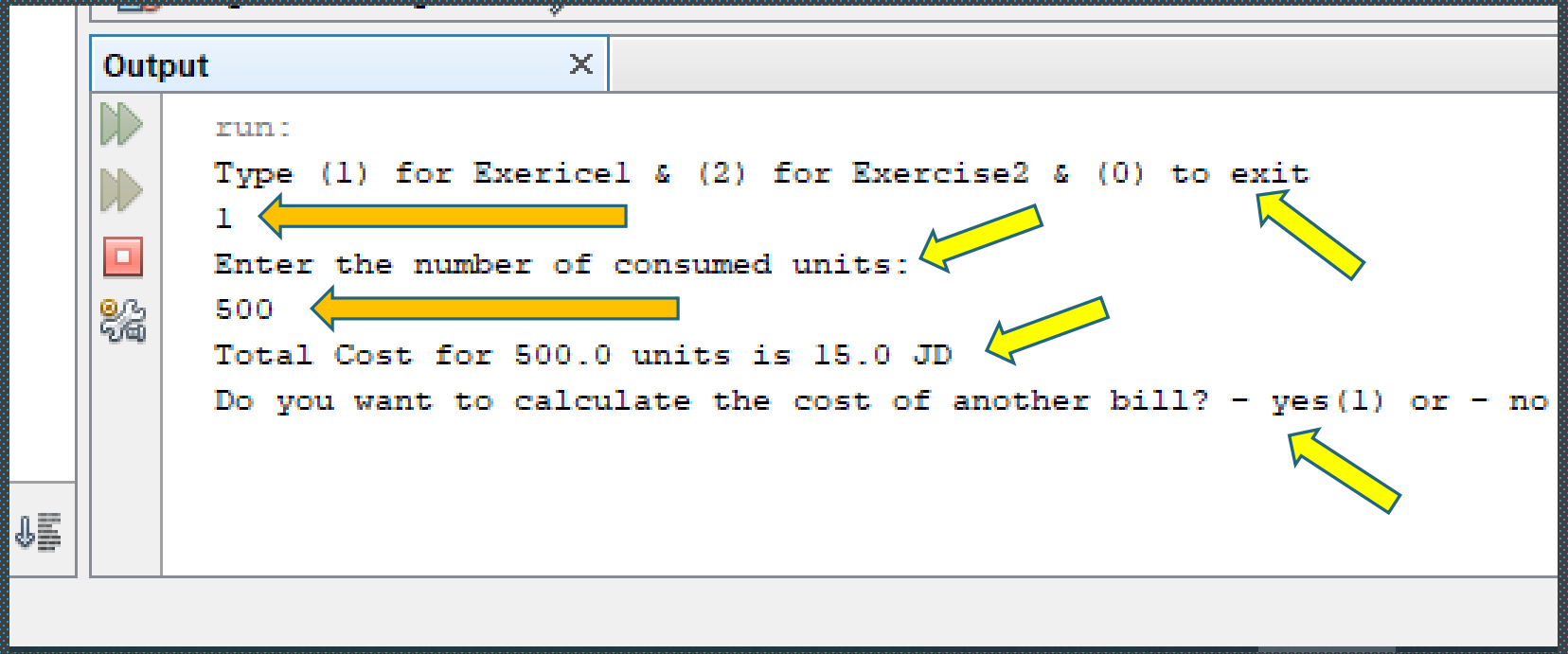


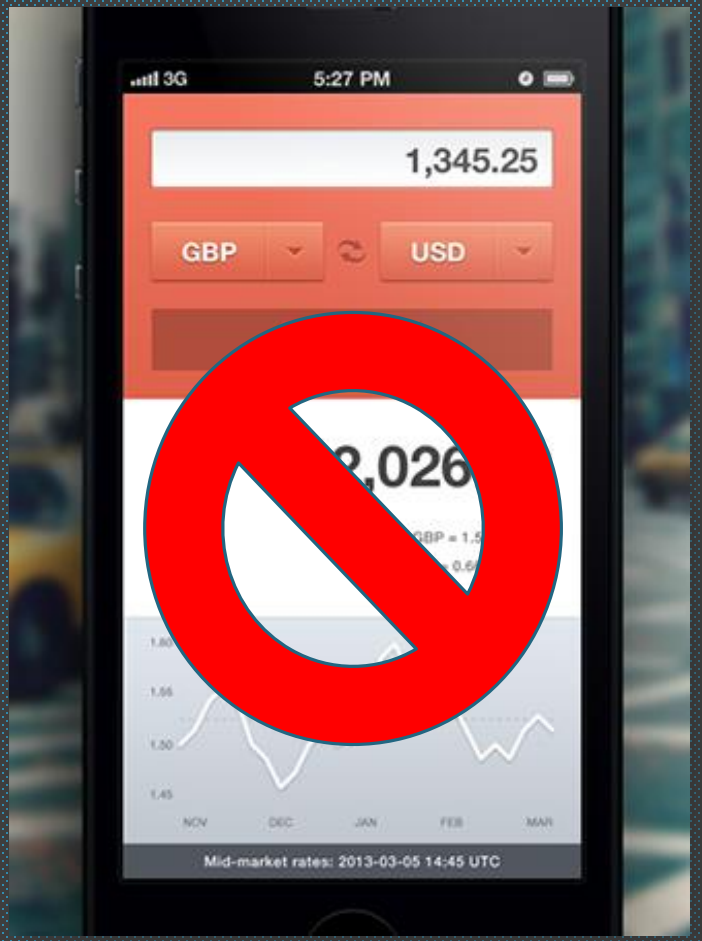
To get/read values from the user

To display massages and results

Input/Output

UNFORTUNATELY NO GUI IN PART 1 OF THIS COURSE





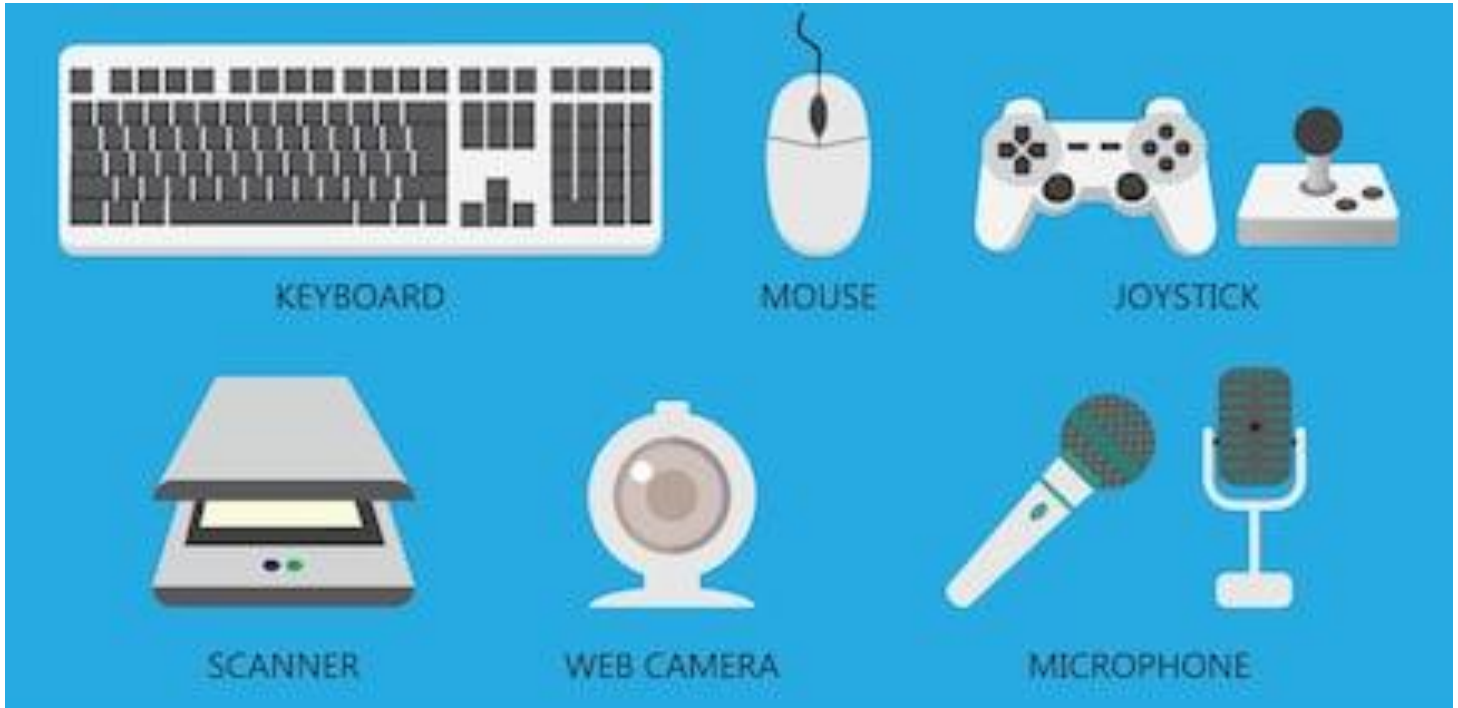
Determine the program's requirements as follows:

1. **Input**: identify the pieces of data that the program needs to read as input.

What information is needed to be processed?

What information do we need from the user?

Decide the names of the variables for those pieces of data, and their data types.



Determine the program's requirements as follows:

2. **Process**: Determine the calculations and/or other instructions that must be performed on the input, or for the output.

What must the program do with the input that it will read?
What data must be processed before output?

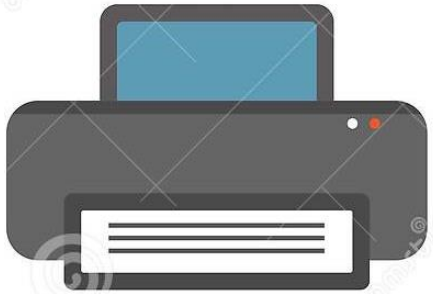
At this time, decide the names and data types of any variables needed to hold the results of calculations.

Determine the program's requirements as follows:

3. **Output**: What output must the program produce?

In most cases, it will be the results of the program's calculations and/or other processes.

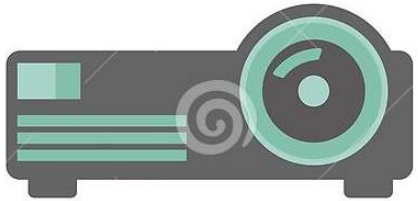
MONITOR



PRINTER

SPEAKER

HEADPHONE



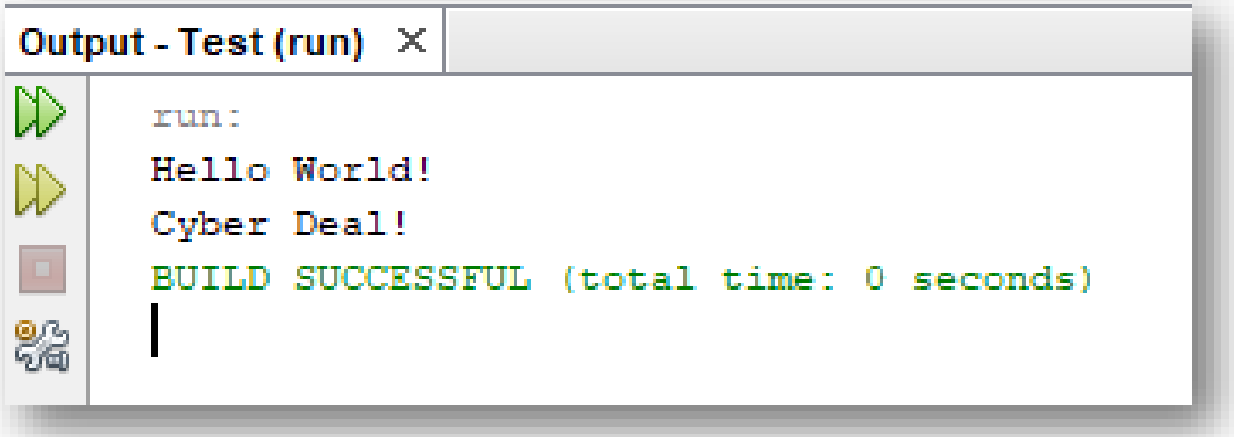
PROJECTOR

Displaying Output on the console

Displaying Output on the console

```
System.out.println("Hello World!");
```

```
System.out.println(“Cyber Deal!");
```



Every program needs a place where it starts.
Java begins execution with the first line of a "main" method which must have the following signature:

public static void main(String[] args)

In Java, every application begins with a class definition.

For now, just remember that, every Java application has a class definition, and the name of class should match the filename in Java.

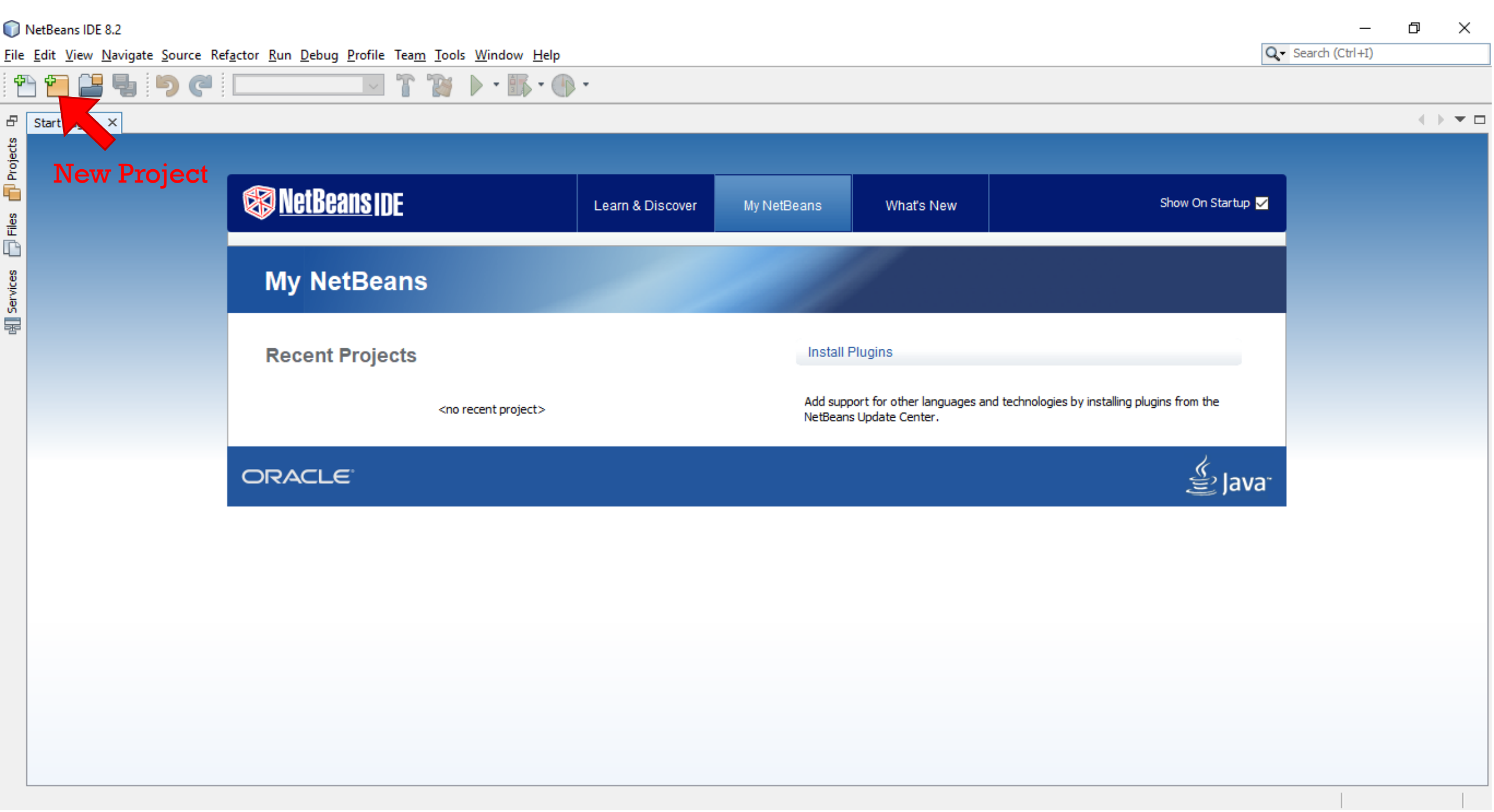
Every application in Java must contain a main method.

The Java compiler starts executing the code from the main method

```
class Yaman {

    public static void main(String[] args) {

        System.out.println("Hello, World!");
        System.out.println("Cyber Deal!");

    }
}
```

** Extra information "First Java Program"

TIME TO CODE


NetBeans IDE 8.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)
Start
Projects
Files
Services
New Project
NetBeans IDE
Learn & Discover
My NetBeans
What's New
Show On Startup
My NetBeans
Recent Projects
<no recent project>
Install Plugins
Add support for other languages and technologies by installing plugins from the NetBeans Update Center.
ORACLE
Java

LETS DO IT ON NETBEANS NOW

JavaApplication14 - NetBeans IDE 8.2

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

Search (Ctrl+I)

<default config>

Proj... Files Services

- JavaApplication14
  - Source Packages
    - javaapplication14
      - JavaApplication14.java
  - Libraries

main - Navigator

Members <empty>

- JavaApplication14
  - main(String[] args)

Start Page | JavaApplication14.java

Source History

```
1  package javaapplication14;
2
3  public class JavaApplication14 {
4
5      public static void main(String[] args) {
6          System.out.println("Yaman");
7      }
8
9  }
10
```

Area to code

javaapplication14.JavaApplication14 > ma

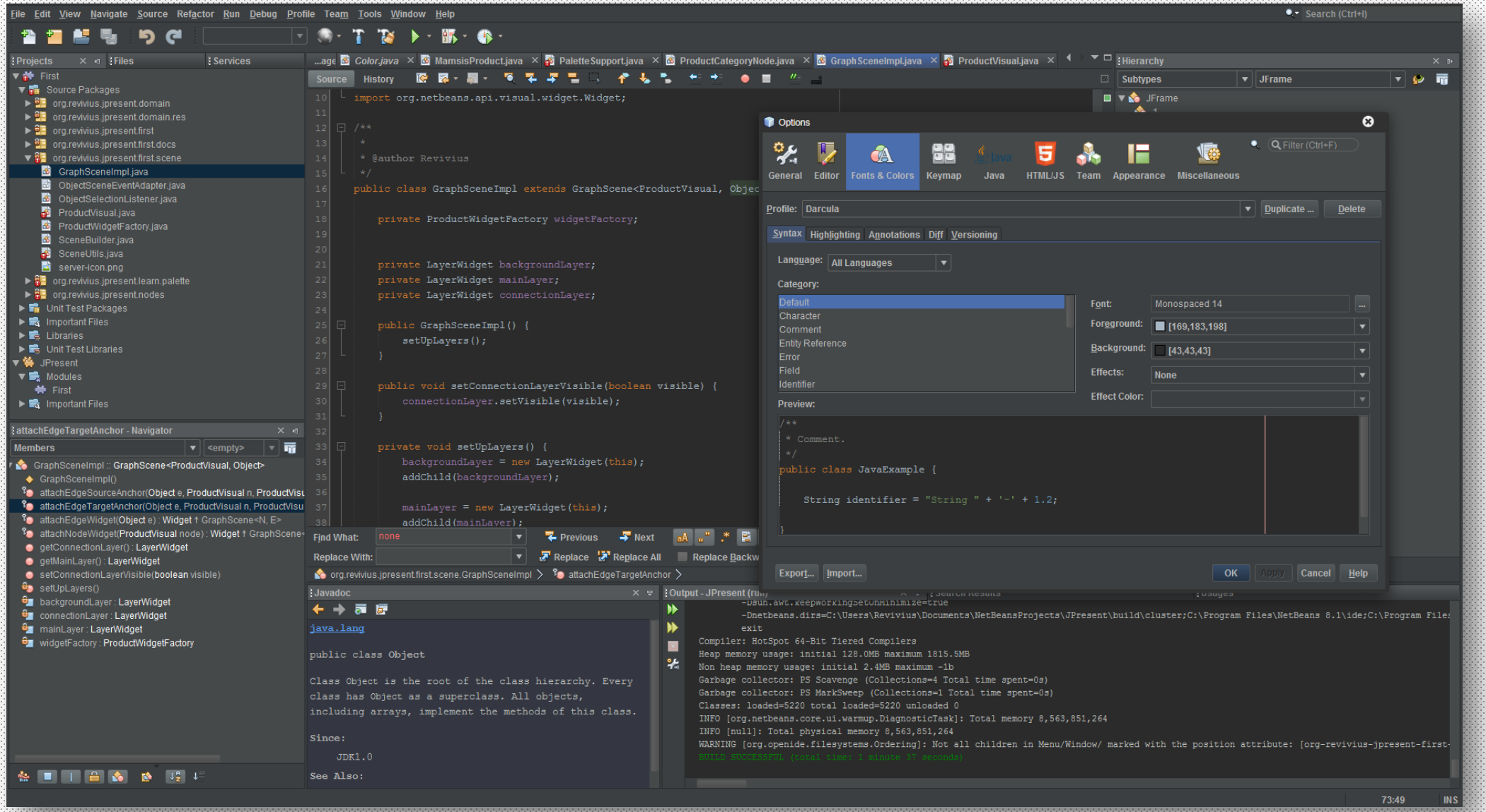
Output will be displayed here

Output - JavaApplication14 (run)

```
run:
Yaman
BUILD SUCCESSFUL (total time: 0 seconds)
```

6:37 INS

Dark Coding Themes Are Available on NetBeans



Dark screen helps to concentrate your eyes longer and helps your brain to keep more attention on the screen.

~ Medium.com // Why Developers Prefer Dark Coding Themes?

Reading Numbers from the Keyboard

```
import java.util.Scanner;

public class HelloWorld {

    public static void main(String[] args) {

        Scanner input = new Scanner(System.in);
        System.out.print("Enter a number: ");

        int number = input.nextInt();

        System.out.println("You entered: " + number);
    }
}
```

Example: How to Print an Integer entered by an user

Reading Numbers from the Keyboard

```
Scanner input = new Scanner(System.in);

System.out.print("Enter an int value: ");
int intValue = input.nextInt();
```

- nextInt(): reads an integer of the int type.
- nextFloat(): reads a number of the float type.
- nextDouble(): reads a number of the double type

Tracing ::: Reading Numbers from the Keyboard

```
 1  import java.util.Scanner;
 2
 3  public class Quotient {
 4    public static void main(String[] args) {
 5      Scanner input = new Scanner(System.in);
 6
 7      // Prompt the user to enter two integers
 8      System.out.print("Enter two integers: ");
 9      int number1 = input.nextInt();
10      int number2 = input.nextInt();
```

Tracing ::: Reading Numbers from the Keyboard

```
 1  import java.util.Scanner;
 2
 3  public class Quotient {
 4    public static void main(String[] args) {
 5      Scanner input = new Scanner(System.in);
 6
 7      // Prompt the user to enter two integers
 8      System.out.print("Enter two integers: ");
 9      int number1 = input.nextInt();
10      int number2 = input.nextInt();
```

The program starts the execution from the main method.

Tracing ::: Reading Numbers from the Keyboard

```
 1  import java.util.Scanner;
 2
 3  public class Quotient {
 4    public static void main(String[] args) {
 5      Scanner input = new Scanner(System.in);
 6
 7      // Prompt the user to enter two integers
 8      System.out.print("Enter two integers: ");
 9      int number1 = input.nextInt();
10      int number2 = input.nextInt();
```

The statement creates an object for performing console input and assigns the object to the reference variable named input.

Tracing ::: Reading Numbers from the Keyboard

```
1  import java.util.Scanner;
2
3  public class Quotient {
4    public static void main(String[] args) {
5      Scanner input = new Scanner(System.in);
6
7      // Prompt the user to enter two integers
8      System.out.print("Enter two integers: ");
9      int number1 = input.nextInt();
10     int number2 = input.nextInt();
```

The statement displays a message to prompt the user for input.

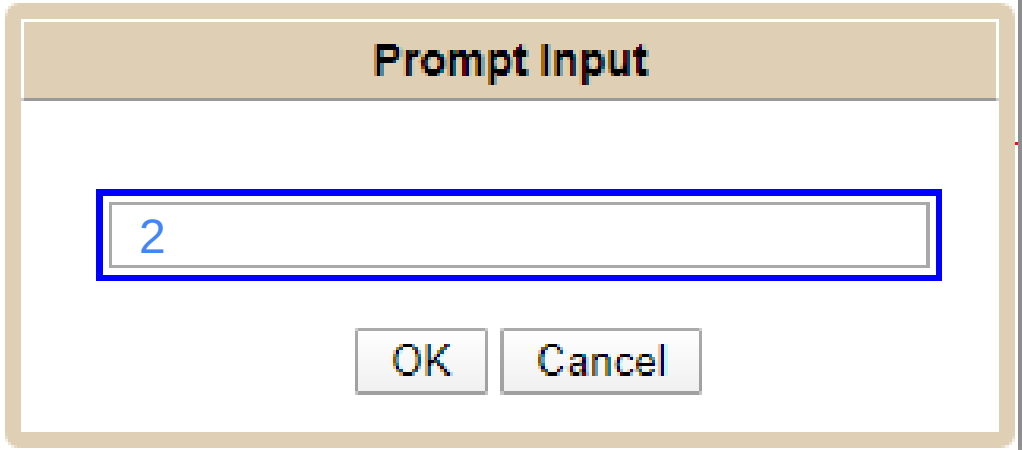
Output

```
Enter two integers:
```

Tracing ::: Reading Numbers from the Keyboard

```
1  import java.util.Scanner;
2
3  public class Quotient {
4    public static void main(String[] args) {
5      Scanner input = new Scanner(System.in);
6
7      // Prompt the user to enter two integers
8      System.out.print("Enter two integers: ");
9      int number1 = input.nextInt();
10     int number2 = input.nextInt();
```

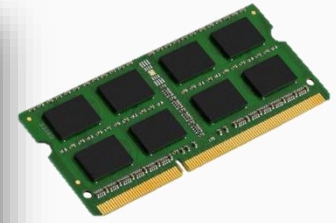
The statement reads an input for number1.



Tracing ::: Reading Numbers from the Keyboard

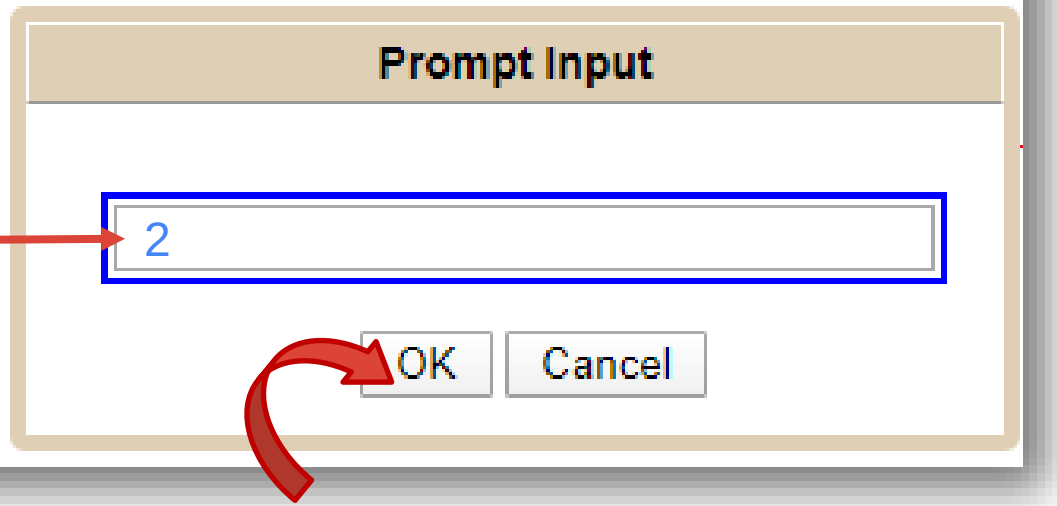
```
1  import java.util.Scanner;
2
3  public class Quotient {
4    public static void main(String[] args) {
5      Scanner input = new Scanner(System.in);
6
7      // Prompt the user to enter two integers
8      System.out.print("Enter two integers: ");
9      int number1 = input.nextInt();
10     int number2 = input.nextInt();
```

Variable Name	Value in Memory
number1	2

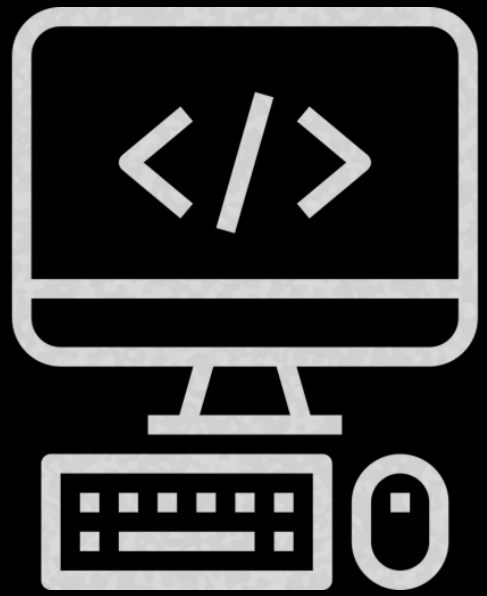


The statement reads an input for number1.

The statement reads an input 2 from the console and assign it to variable number1.



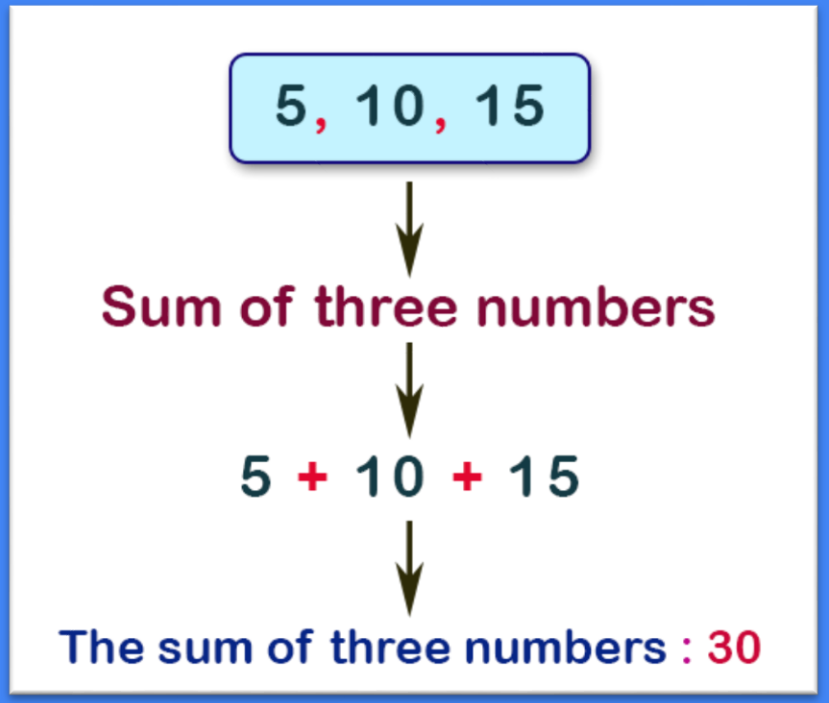
And the same thing for ‘number2’



Coding is “ART”

You sell a line of code the same way you sell a painter or an artist a brush, or anything else contributing to the final product of whatever "the bigger picture" may be

Program to find and print the sum of three integers
(your program should begin reading three numbers from the user)



SUM / TOTAL

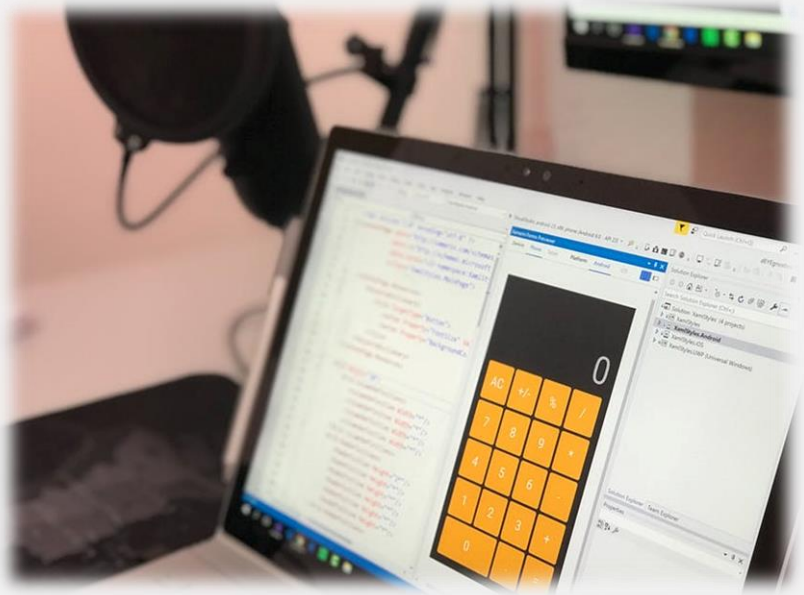
Summation is the operation of adding multiple numbers together

the result is their sum or total

EXAMPLES:

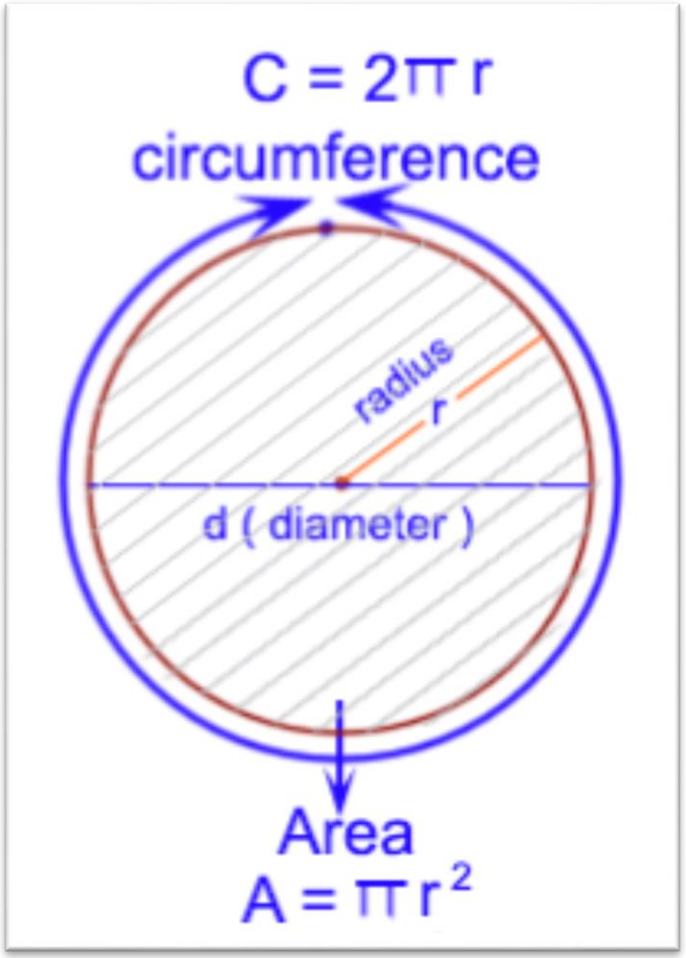
To find the sum of all entries:
e.g. Sum of all class grades
e.g. Sum of the temperatures for this year
e.g. Sum of all salaries in the department
e.g. Sum of all salaries in the whole company

~~**To find the sum of some entries:**~~
~~e.g. Sum of grades that are larger than 70~~



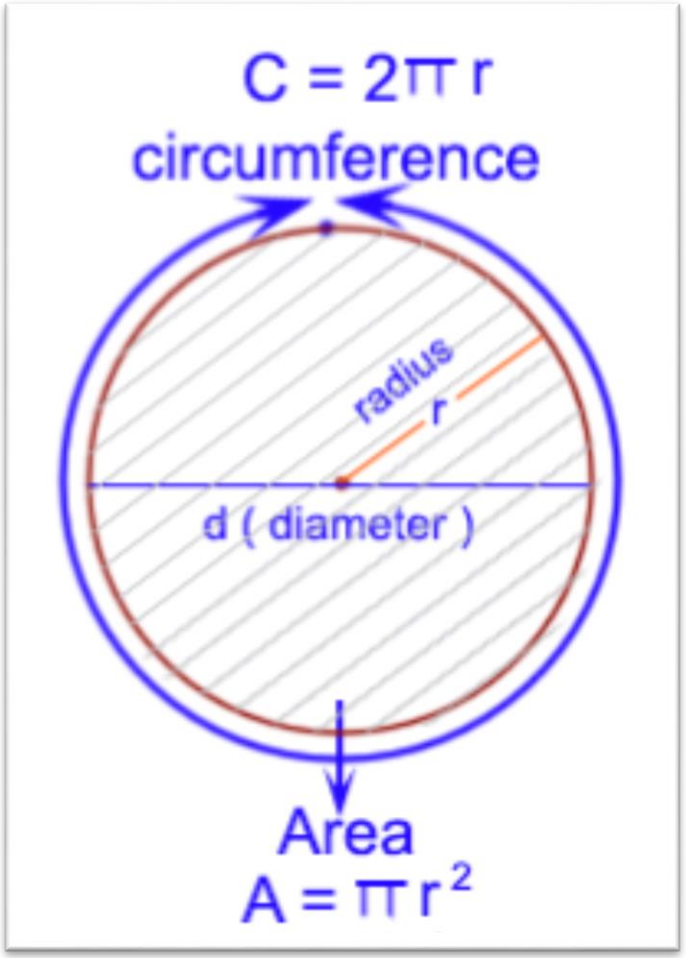
e.g. Sum of grades :
TOTALGRADES = grade1 + grade2 + grade3 … + gradeN

Write a Java program to compute the perimeter and area of a circle with a radius of x inches.



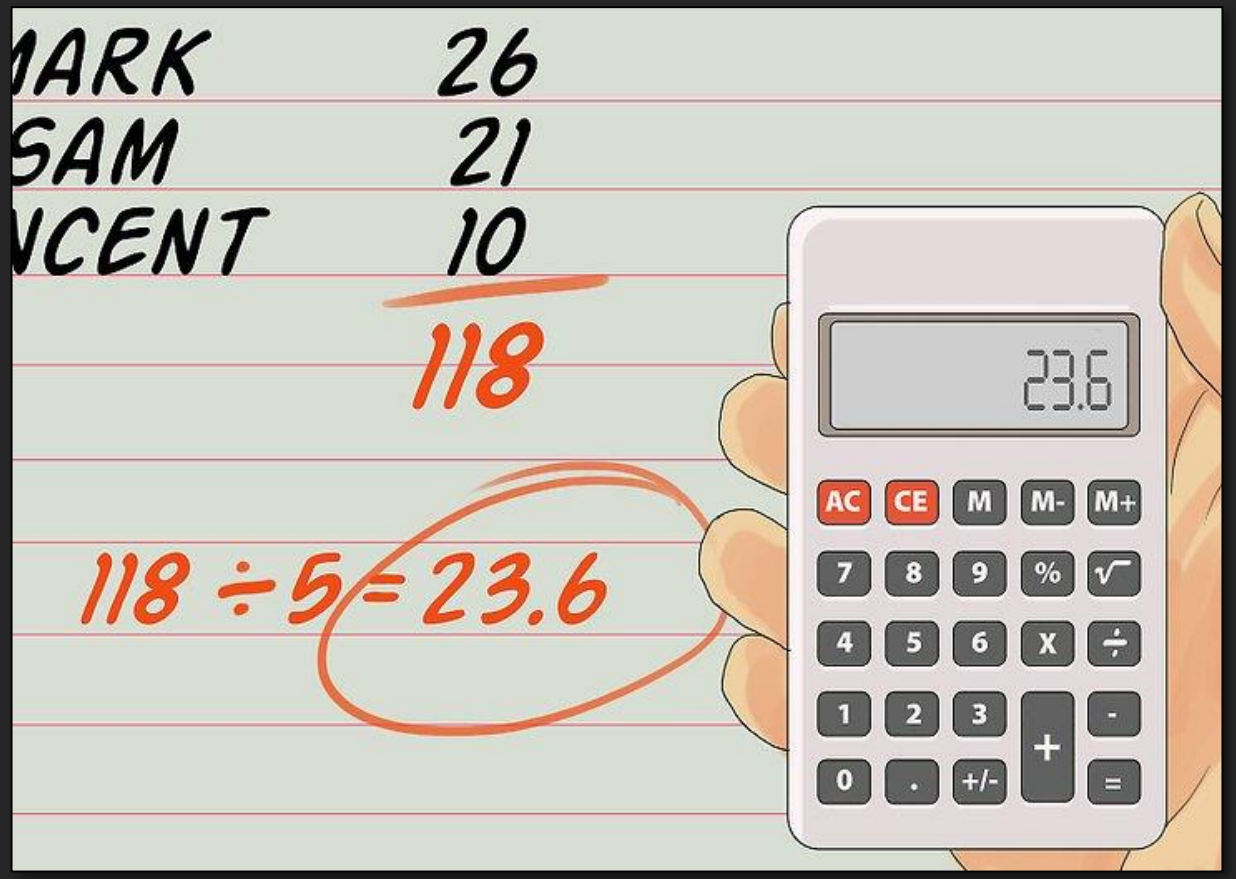
1) Input radius of circle from user. Store it in a variable
2) Apply the formulas to calculate the perimeter and area.
3) Print the results

Write a Java program to compute the perimeter and area of a circle with a radius of x inches.




```java
import java.util.Scanner;
public class Test {
  public static void main(String[] args) {
    // Create a Scanner object
    Scanner input = new Scanner(System.in);

    // Prompt the user to enter a radius
    System.out.print("Enter a number for radius: ");
    double radius = input.nextDouble();

    // Compute area
    double area = radius * radius * 3.14159

    // Display result
    System.out.println("The are is " + area);
  }
}
```

Find and print/display the mean (average) of 4 numbers.



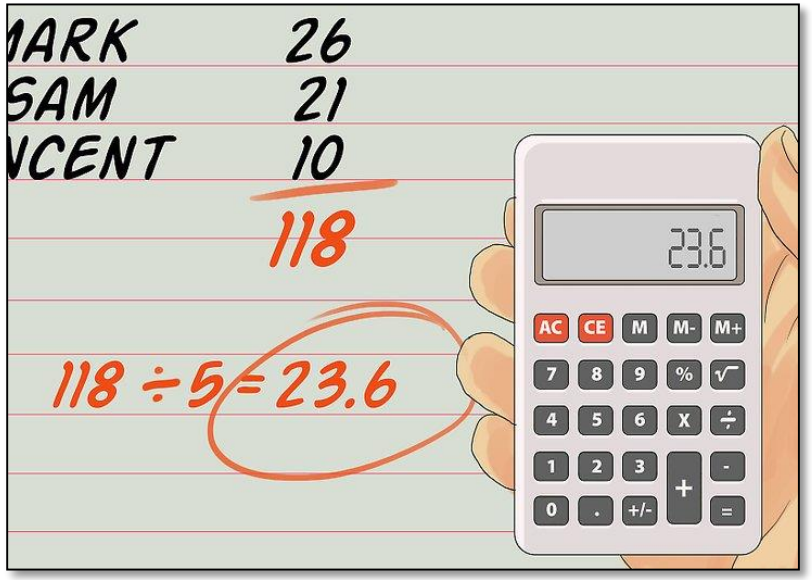
Average of all entries

average = sum of all entries / number of entries

- Sum is calculated first

How to calculate the average?

Step 1 → Start collecting values (using a input statement or maybe its pre-initialized)
Step 2 → Add all values (entries) together using "sum" or "total"
Step 3 → Divide the output of Step 2 with total number of entries
Step 4 → Display the output of Step 3



Read an integer variable named “inHours” which represents time in hours, then save the result in minutes in another variable named “inMinutes”

```
 int inHours , inMinutes;

Scanner input = new Scanner(System.in);
System.out.println("Enter Number of Hourse: ");
inHours = input.nextInt();

inMinutes = inHours * 60;
System.out.println(" In minutes: "+ inMinutes );
```

Multiple Choice:

1. A(n) ____ is a set of well-defined logical steps that must be taken to perform a task.
a. algorithm / b. plan of action / c. logic schedule

2. A __ is any hypothetical person that is using a program and providing input for it.
a. designer **b. user** c. guinea pig d. test subject

3. A ___ is a storage location in memory that is represented by a name.
a. variable / b. register / c. RAM slot / d. byte

Multiple Choice:

4. A(n) ___ is a message that tells (or asks) the user to enter a specific value.
a. inquiry / b. input statement / c. directive / **d. prompt**

5. A(n) ____ sets a variable to a specified value.
a. declaration / **b. assignment statement** / c. math expression

6. n the expression 12 + 7, the values on the right and left of the + symbol are called ____.
a. operands / b. operators / c. arguments / d. math expressions

7. Assigning a value to a variable in a declaration statement is called ____.
a. allocation / **b. initialization** / c. certification

BOOLEANS

TRUE? FALSE?

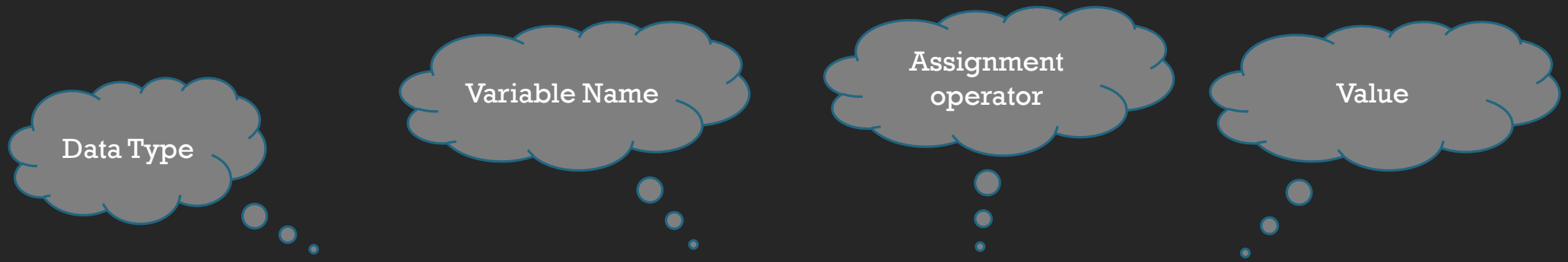
BOOLEANS

YES / NO

ON / OFF

TRUE / FALSE

BOOLEANS




```
boolean yaman = true;
```

Not so useful? ☹
Then lets compare stuff

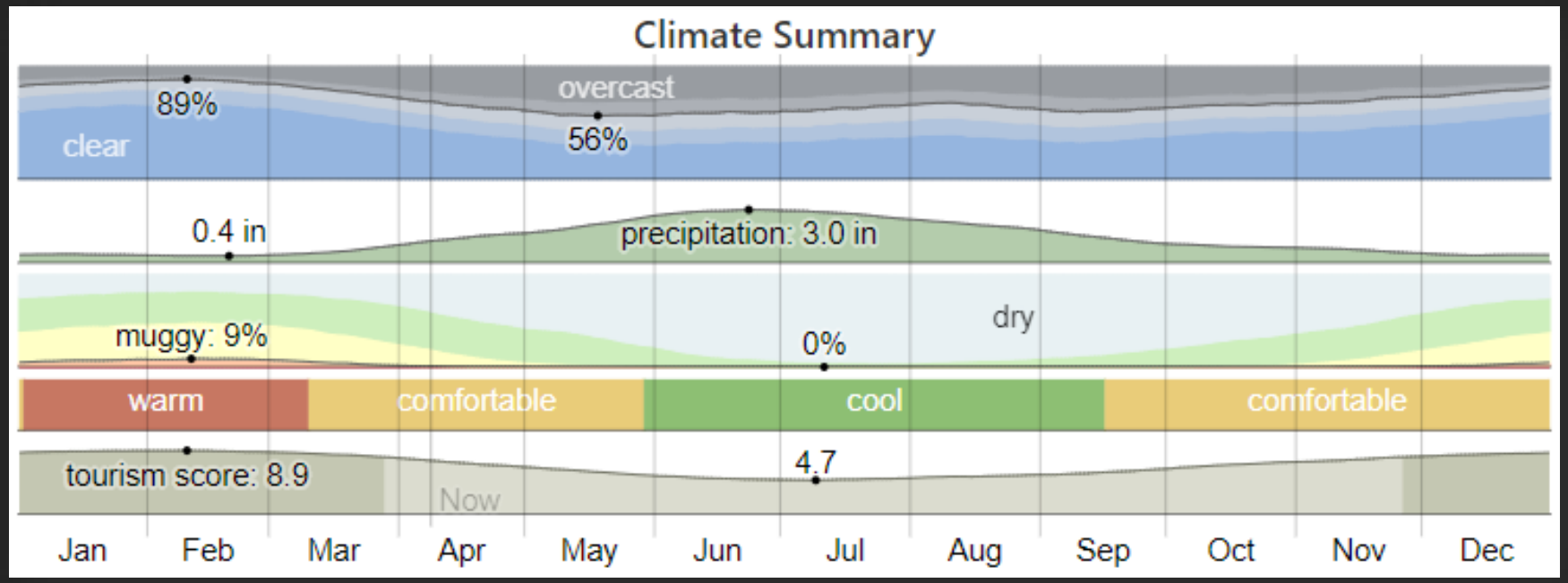
COMPARING OPERATORS

Operator	Description
==	Returns true if the expression on the left evaluates to the same value as the expression on the right.
!=	Returns true if the expression on the left does not evaluate to the same value as the expression on the right.
<	Returns true if the expression on the left evaluates to a value that is less than the value of the expression on the right.
<=	Returns true if the expression on the left evaluates to a value that is less than or equal to the expression on the right.
>	Returns true if the expression on the left evaluates to a value that is greater than the value of the expression on the right.
>=	Returns true if the expression on the left evaluates to a value that is greater than or equal to the expression on the right.

SO BASICALLY WE COMPARE STUFF

E.g. A program for a automated air condition that checks if the weather is hot or cold.
OR maybe a program that classifies each month of the year if it was hot or cold.

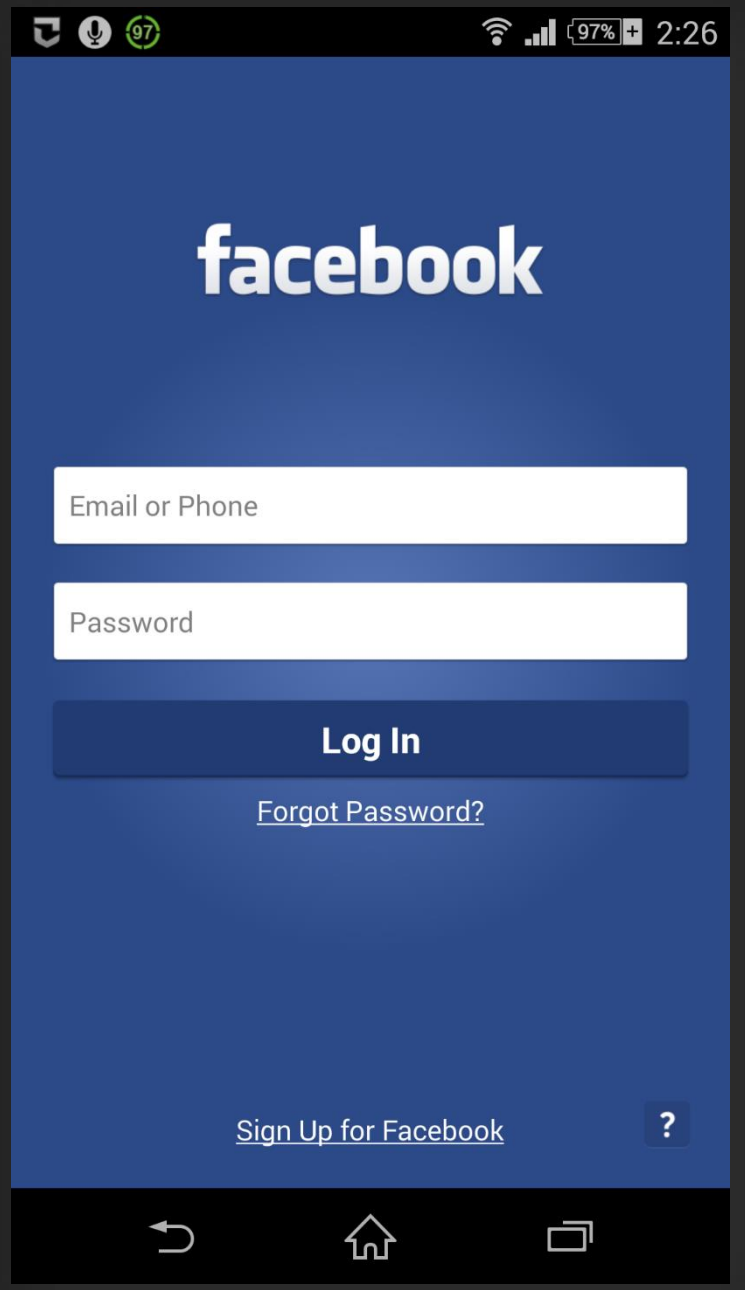
```
boolean isCold = temp < 12;

boolean isHot = temp > 35;
```



https://weatherspark.com/

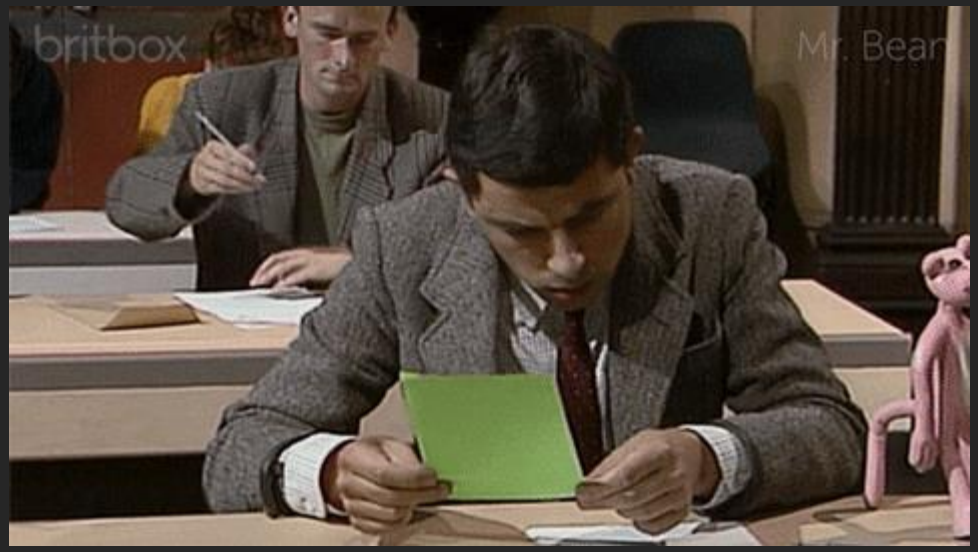
A program that checks if the user has entered the correct password

```
int password = 3322;

boolean isTrue = (password == 3322);
```



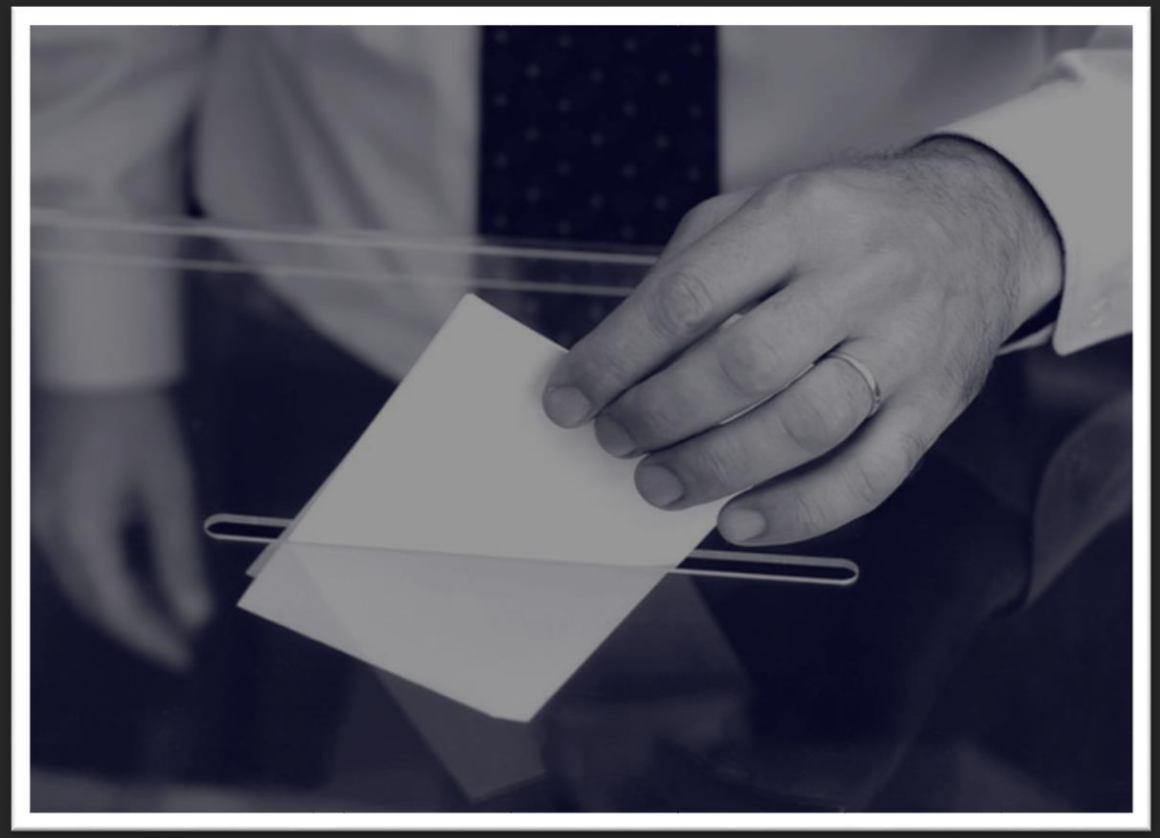
A program that checks if the student passed the exam

```
int grade = 85;
boolean passed = grade >= 50;
```



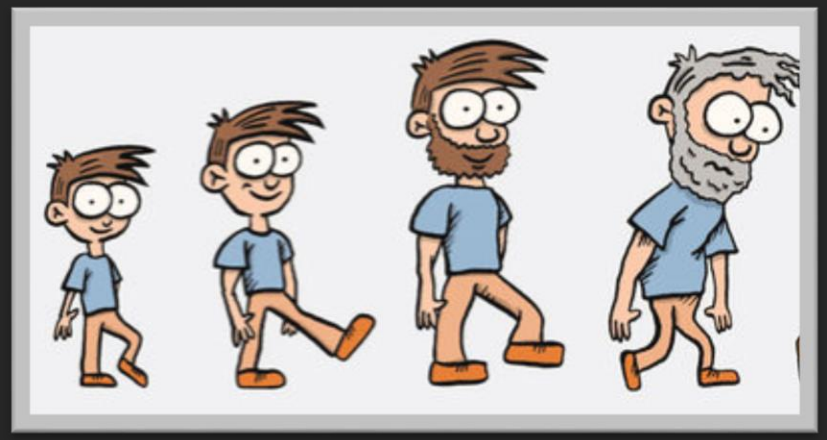
A program that checks if the candidate eligible for elections

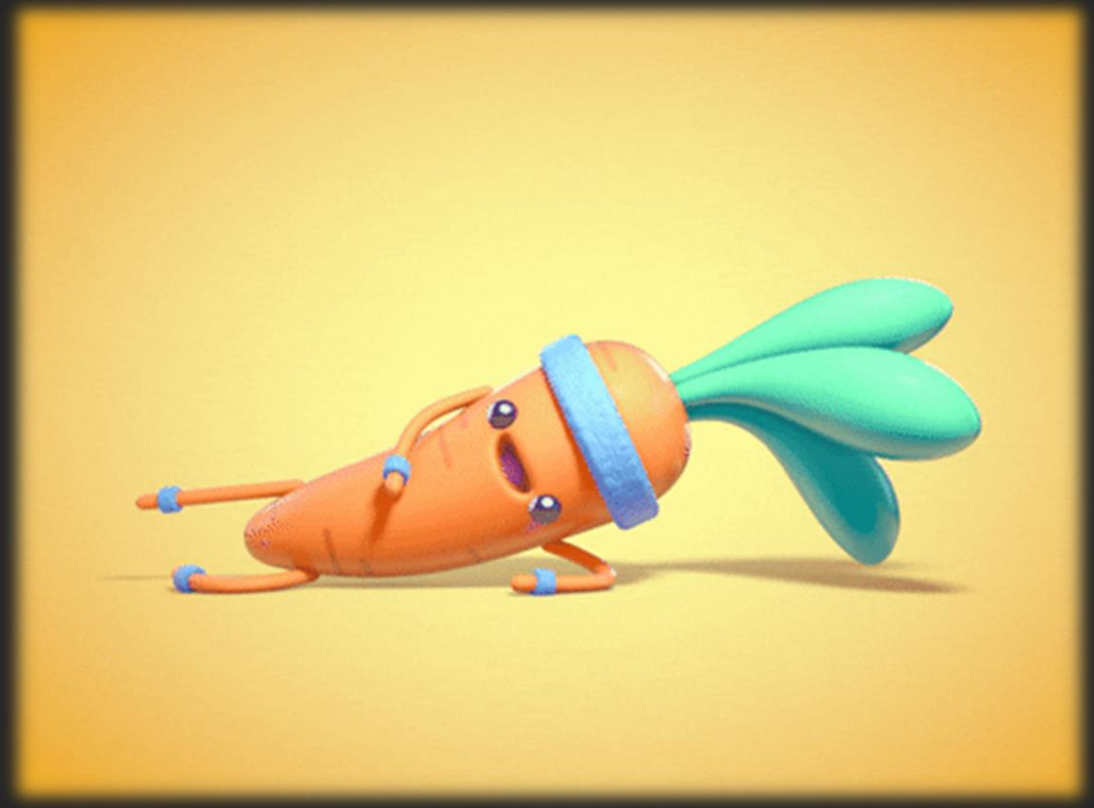
```
int age = 22;
boolean isEligible = age>=30;
```



A program that checks if the user is in his twenties

```
int age = 22;
boolean isInTwenties =
```





This takes us to the 3
basic operations on Booleans,

Basic operations on Booleans

!

Not (Flips the output)
!= (Not Equal to) >>> Only 1, 2, or few conditions should not be met
e.g. When we have (A/B) then B should be != 0 (divide by zero)

&&

And
>>> All conditions must be true
e.g. To login your "username" **AND your "password" should be correct**

||

Or
>>> At least one condition must be true
e.g. You can submit your homework either on youtube **OR** on facebook
e.g. To finish the payment you can pay by a credit card **OR** using cash

Not
(Flips the output)


```
int grade = 85;
boolean passed = grade >= 50;

boolean failed = !passed;
```

F

COMPLEX CONDITION

Condition 1 && Condition 2

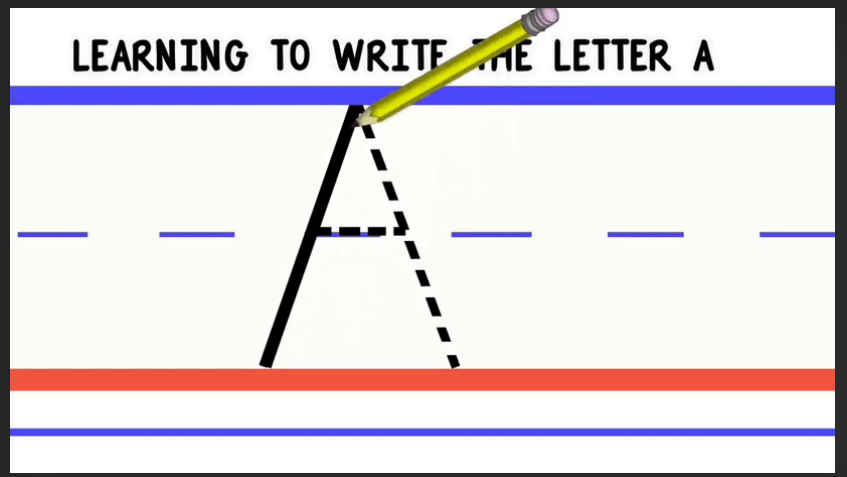
Condition 1 || Condition 2

A program that checks if the letter is uppercase or not

```
char letter = 'B';
boolean isUpper = (letter >= 'A') && (letter <= 'Z');
```

You should always use a capital letter in the following situations:

1. In the names of people, places, or related words. Use a capital letter when you are writing the names of people, places, and words relating to them:
2. At the beginning of a sentence. ...
3. In the titles of books, films, organizations, etc. ...
4. In abbreviations.



Challenge: Check if the “char letter” is not a alphabetical character

E.g. A program to check if the weather is good for a picnic

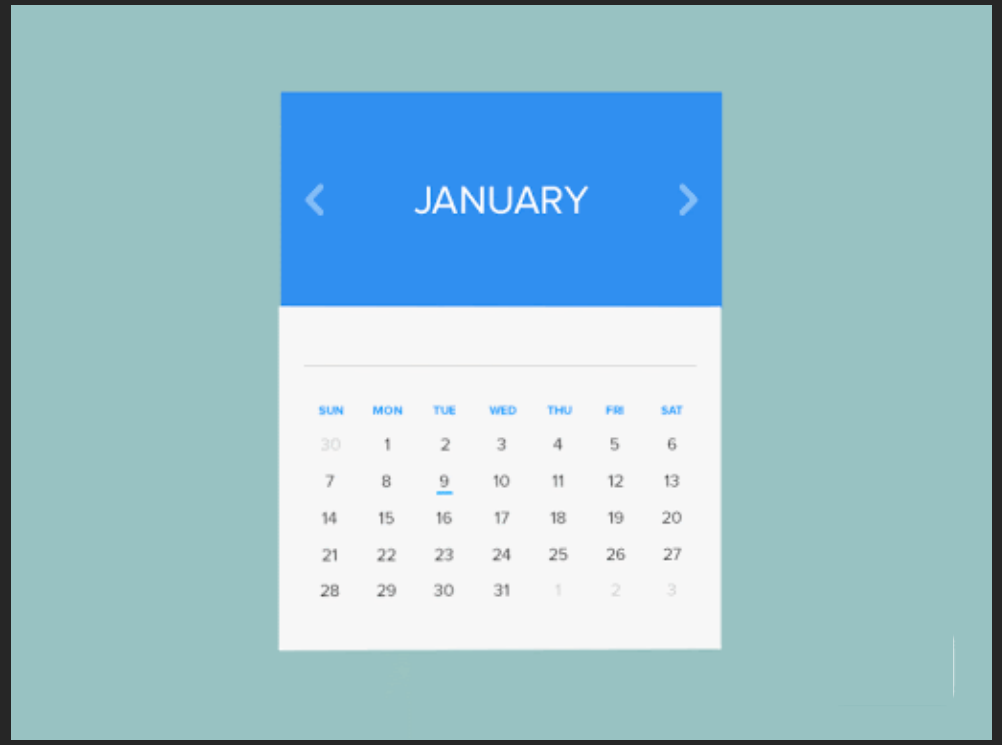
```
int temp = 22;
boolean isGood = (temp >= 12) && (temp <=35);
```

what if we do the following?

```
boolean isGood = (temp > 12) && (temp < 35);
```

A program that checks if the month is valid

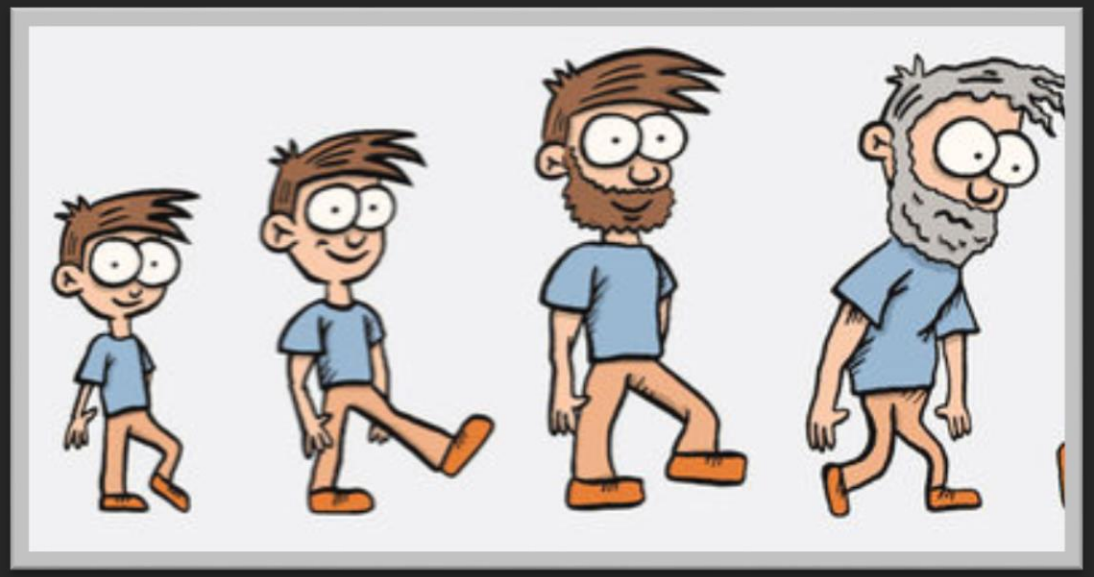
```
int month = 5;
isValidMonth = (month >= 1) && (month <= 12)
```



Challenge: Check if the “int day” is a existing day in June

A program that checks if the user is in his twenties

```
int age = 22;
boolean isInTwenties = age>=20 && age<30
```



Free shipping on orders that exceed 100$ or for premium members


```
int cart = 40;
boolean premium = true;
boolean takeDiscount =  cart>100 || premium;
```

hasInsurance

```
boolean isEmployee = true;
boolean isOver65 = false;
boolean hasInsurance = isEmployee || isOver65;
```

```
boolean isFuelFull = true;
boolean isTiresOk= true;
boolean isSafe = isFuelFull && isTiresOk;

==================================
```

We will talk much more about Booleans when we reach strings

Constants in Java *(final)*

a named location that can store a value (like a variable) ,
but the value cannot be changed after it has been defined
at the beginning of the program .

(cannot be changed during the program's execution)

e.g.
PI,
GRAVITY,
SALES_TAX,
SPEED_LIMIT

Constants in Java (final)

Assume that the following statement appears in a accounting program that calculates data pertaining to tax:

totalTax = balance * 0.069

In such a program, two potential problems arise.

First, it is not clear to anyone other than the original programmer what 0.069 is.
It appears to be an tax rate, but in some situations there are fees associated with payments.
The purpose of this statement can't be determined without painstakingly checking the rest of the program

The second problem occurs if this number is used in other calculations throughout the program and must be changed periodically.

what if the rate changes from 6.9 percent to 7.2 percent?
The programmer would have to search through the source code for every occurrence of the number.

The following is an example of how we will declare named constants:
final double TAX_RATE = 0.069

The following statement:
totalTax = balance * 0.069
can be changed to read
amount = balance * TAX_RATE

Constants in Java (final)

Notice that the declaration looks a lot like a variable declaration, except that we use the word final

Also, notice that the name of the constant is written in all uppercase letters. This is a standard practice in most programming languages because it makes named constants easily distinguishable from regular variable names.

An initialization value must be given when declaring a named constant.