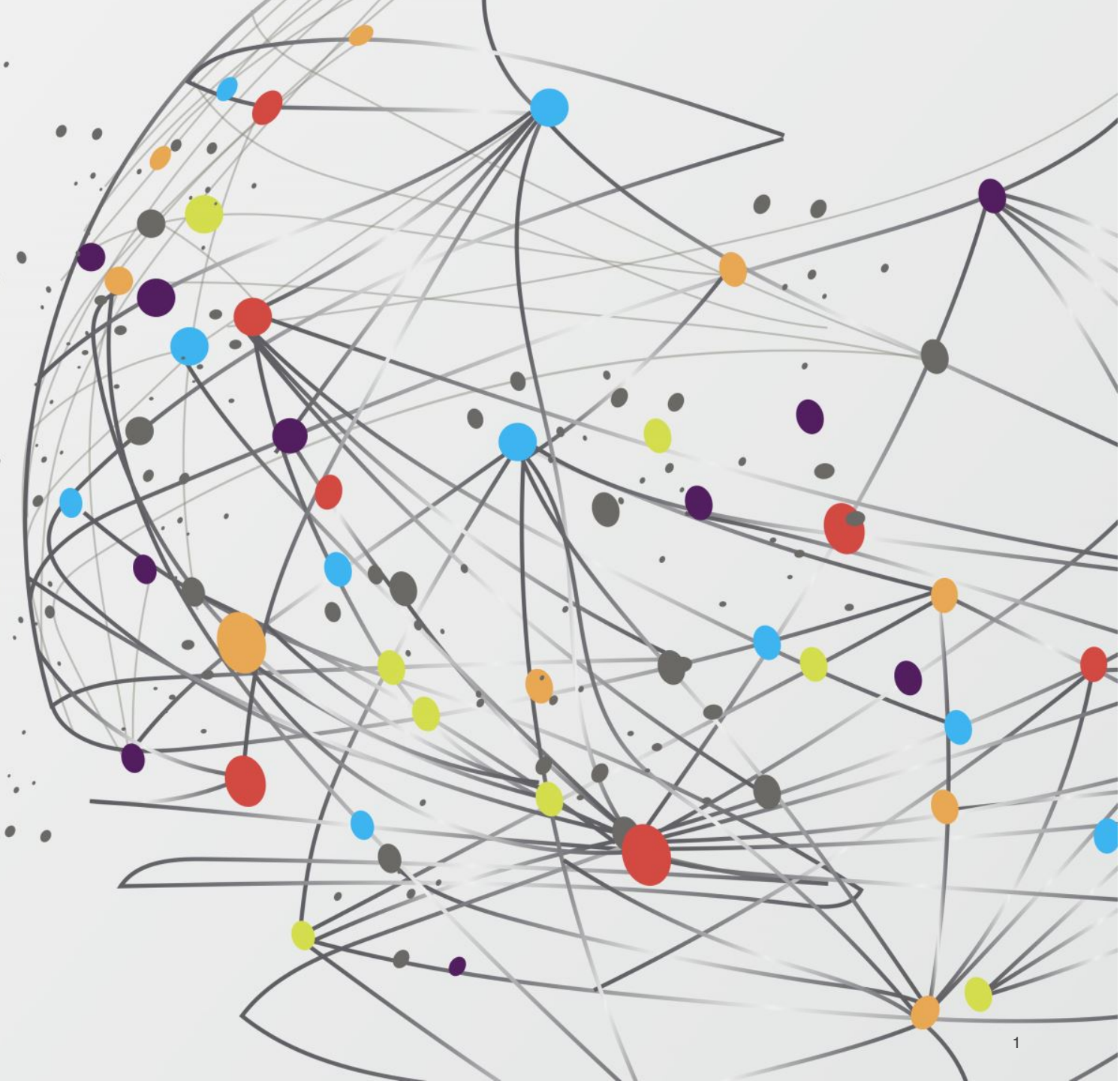
WEEK 1



NEWTON SCHOOL

INTRODUCTION

JAVA/JDK INTRO AND SETUP

- You can install JDK from oracle.com https://www.oracle.com/java/technologies/downloads/#java8-windows
- You can install community version of IntelliJ https://www.jetbrains.com/idea/download/#section=mac . It provides a development environment.
- Java is one of the most popular programming languages because it is used in various tech fields like app development, web development, client-server applications, etc.
- Java is an object-oriented programming language developed by Sun Microsystems of the USA in 1991.
- It was originally called Oak by James Goslin. He was one of the inventors of Java.
- Java = Purely Object-Oriented.

HOW JAVA WORKS

- Source code compiled to byte code i.e. .class file . It is platform independent.
- Bytecodes are non-runnable codes and rely on the availability of an interpreter to execute and thus the JVM comes into play.
- Bytecode is essentially the machine level language(assembly language) which runs on the Java Virtual Machine.
- Byte code interpreted to machine code with the help of Java virtual machine so that it can run in Windows, Linux or any other Operating system.
- JDK = JVM + JRE -> Java Development Kit -> Collection of tools used for developing and running Java programs.
- JRE -> Java Runtime Environment -> Helps in executing programs developed in Java.

VARIABLES

- A variable is a container which holds the value while the program is executed. A variable is assigned with a data type.
- Variable is a name of memory location. There are three types of variables in java: local, instance and static.
- A variable is the name of a reserved area allocated in memory. Its value can be changed.
- Int sum=50; int-> datatype, sum-> variable
- Local variable-> method variable.
- Instance Variable -> Class variable -> Instance of a class specific
- Static variable -> Sharable among all instances, Memory allocation for static variables happens only once when the class is loaded in the memory.

DATATYPES

- Data types specify the different sizes and values that can be stored in the variable. There are two types of data types in Java:

- **Primitive data types - basic:** The primitive data types include boolean, char, byte, short, int, long, float and double.

- **Non-primitive data types:** The non-primitive data types include Classes, Strings, Arrays etc

OPERATORS

- Are symbols that is used to perform operations. For example: +, -, *, / etc. Operand is the one on which operation is performed.
- There are many types of operators in Java which are given below:
  - Unary Operator, Eg ++, - -
  - Arithmetic Operator, Eg +, -
  - Ternary Operator ? :
  - Assignment Operator = etc

HOW TO APPROACH ANY PROBLEM ??

- Understand the problem.
- Analyse the requirement.
- Analyse the solution.
- Think of the better solution.
- Code it.
- Test it.
- Check all the base cases that should get covered.
- You are all set !

UNDERSTAND THE PROBLEM

1. Deep dive into what's already given into the question.
2. What is Asked ?
3. Understand the sample input/output given.
4. Write few test cases and verify your understanding.

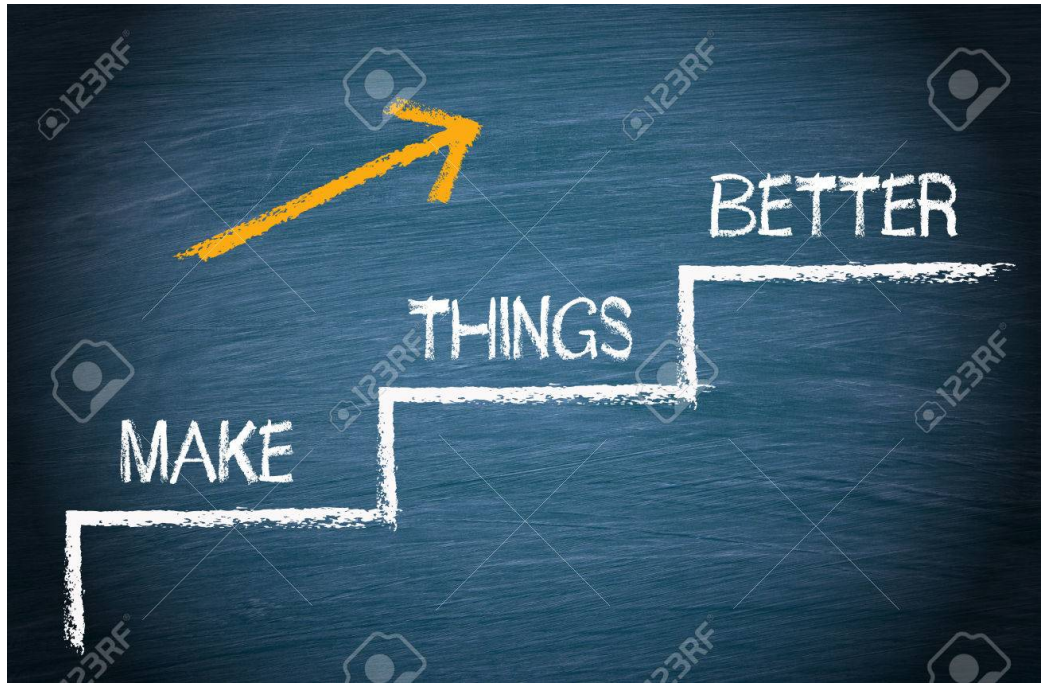
ANALYSE THE PROBLEM

1. What are the implications?
2. What are the key sub tasks inside the problem.
3. Checking the blocking task.
4. Simplify using more test cases.

THINK OF A SOLUTION

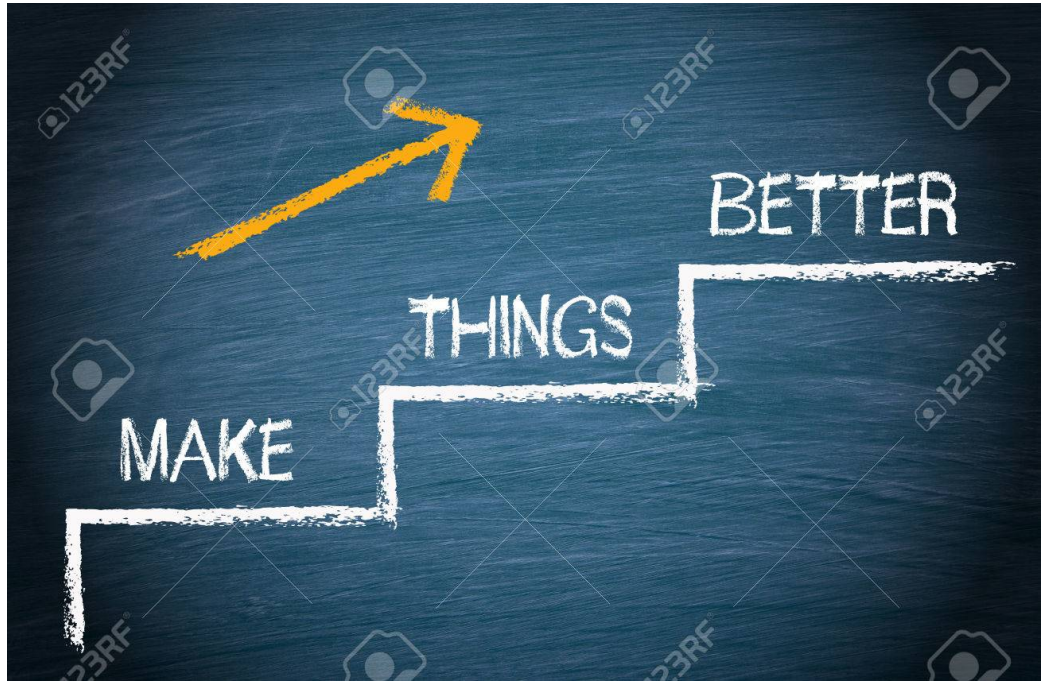
1. Think of a Brute Force solution
2. Think of applying basic data structures
3. Sorting/Searching/Hashing/Heaps/Queue/Stack…
4. Break the problem into smaller sub problems. Eg. swap / power functions.

THINK OF A BETTER SOLUTION



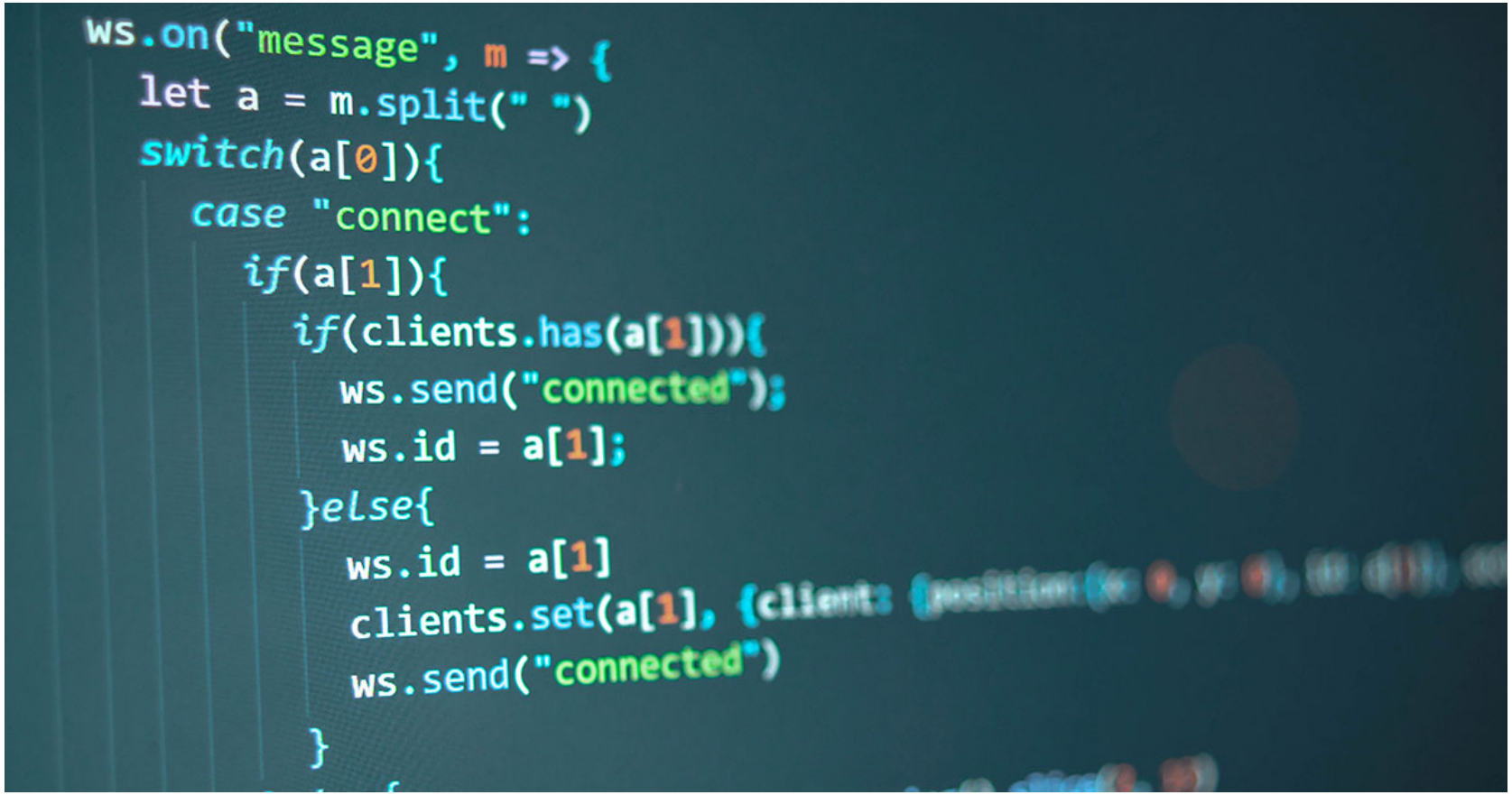
1. Think of reducing the time complexity.
2. Reduce Space complexity.
3. Think of trade off between both.
4. Think a little about similar problem you have solved in past.
5. Start every time from scratch.
6. Don’t stop until you reach the best.
7. You will get it with more practice and experience.

WRITE PSEUDO CODE



1. Pseudo code is nothing but a pen paper version of your solution in your known language.
2. Run the test cases over your pseudo code.
3. Test it.
4. Try to make it Perfect.

CODE IT FOR YOUR COMPILER TO UNDERSTAND



I/P O/P AND EXCEPTIONS

- A **Java I/O** (Input and Output) is used *to process the input* and *produce the output*.
- Java uses the concept of a stream to make I/O operation fast. The java.io package contains all the classes required for input and output operations.
- We can perform file handling in Java by Java I/O API.
- A stream is a sequence of data. In Java, a stream is composed of bytes. It's called a stream because it is like a stream of water that continues to flow.
- An exception (or exceptional event) is a problem that arises during the execution of a program. When an **Exception** occurs the normal flow of the program is disrupted and the program/Application terminates abnormally, which is not recommended, therefore, these exceptions are to be handled.

I/P O/P AND EXCEPTIONS

- System.out.println("simple message");
- System.err.println("error message");

- **int** i=System.in.read();//returns ASCII code of 1st character
- System.out.println((**char**)i);//will print the character
- Java application uses an output stream to write data to a destination; it may be a file, an array, peripheral device or socket.
- Java application uses an input stream to read data from a source; it may be a file, an array, peripheral device or socket.

I/P O/P AND EXCEPTIONS

- Java **Scanner class** allows the user to take input from the console. It belongs to **java.util** package. It is used to read the input of primitive types like int, double, long, short, float, and byte. It is the easiest way to read input in Java program.
- Scanner sc= **new** Scanner(System.in); //System.in is a standard input stream
- System.out.print("Enter first number- ");
- **int** a= sc.nextInt();
- String str= sc.nextLine(); // reads string

CONTROL STRUCTURES

- If else If
- The condition of the If statement gives a Boolean value, either true or false.
- **if** (count > 2) {

```
    System.out.println("Count is higher than 2");
} else {
    System.out.println("Count is lower or equal to 2");
}
```

- Switch case
- If we have multiple cases to choose from, we can use a *switch* statement.
- Three or more *if/else* statements can be hard to read. As one of the possible workarounds, we can use *switch,* as seen below.
- **int** count = 3;

```
switch (count) {
case 0:
    System.out.println("Count is equal to 0");
    break;
case 1:
    System.out.println("Count is equal to 1");
    break;
default:
    System.out.println("Count is either negative, or higher than 1");
    break;
}
```

- Ternary Operator
- We can use it as a shorthand expression that works like an *if/else* statement.
- Eg: `System.out.println(count > 2 ? "Count is higher than 2" : "Count is lower or equal than 2”);`
- While ternary can be a great way to make our code more readable, it isn't always a good substitute for *if/else.*

LOOPS

- In programming, sometimes we need to execute the block of code repeatedly while some condition evaluates to true.
- However, loop statements are used to execute the set of instructions in a repeated order. The execution of the set of instructions depends upon a particular condition.
- In Java, we have three types of loops that execute similarly. However, there are differences in their syntax and condition checking time.

LOOPS

- For

```
1.  int sum = 0;
2.  for(int j = 1; j<=10; j++) {
3.  sum = sum + j;
4.  }
```

- For each

```
5.  String[] names = {"Java","C","C++"};
6.  System.out.println("Printing the content o
    f the array names:\n");
7.  for(String name:names) {
8.  System.out.println(name);
9.  }
```

- While

```
1.  int i = 0;
2.  System.out.println("Printing the list of fi
    rst 10 even numbers \n");

3.  while(i<=10) {
4.  System.out.println(i);
5.  i = i + 2;
6.  }
```

- Do While

```
1.  int i = 0;
2.  System.out.println("Printing the list of fi
    rst 10 even numbers \n");

4.  do {
5.  System.out.println(i);
6.  i = i + 2;
7.  }while(i<=10);
```

FUNCTIONS

- A **method or function** is a way to perform some task.
- A **method** is a block of code or collection of statements or a set of code grouped together to perform a certain task or operation.
- It is used to achieve the **reusability** of code.
- It also provides the **easy modification** and **readability** of code, just by adding or removing a chunk of code.
- The method is executed only when we call or invoke it.
- The most important method in Java is the **main()** method which is the starting point for JVM to start execution of a Java program. Without the main() method, JVM will not execute the program.

FUNCTIONS

- The method declaration provides information about method attributes, such as visibility, return-type, name, and arguments.
- Example: public/private/.. int sum(int a, int b){ // body }
- There are two types of methods in Java:
  - Predefined Method Eg: print(), sqrt()
  - User-defined Method

RECURSION

- Recursion in java is a process in which a method calls itself continuously.
- Function in java that calls itself is called recursive method.
- It makes the code compact but complex to understand.
- Example : program for factorial

RECURSION

```
1.  public class Recursion{

3.      static int factorial(int n){
4.          if (n == 1)
5.            return 1;
6.          else
7.            return(n * factorial(n-1));
8.      }
9.
10. public static void main(String[] args) {
11. System.out.println("Factorial of 5 is: "+factorial(5));
12. }

14. }
```

```
factorial(5)
   factorial(4)
      factorial(3)
         factorial(2)
            factorial(1)
               return 1
            return 2*1 = 2
         return 3*2 = 6
      return 4*6 = 24
   return 5*24 = 120
```

QUESTIONS

THANKS