Introduction to Computer Science
Reichman University

Recitation 1

Recitation 1

Administration

Before We Start

- Make sure to be updated in the Moodle Forum
  - The forum is conducted in English
- Read the “From Zero to Code” in Moodle and install course tools on your computer.
- Read the “Java Code Style Guidelines” in Moodle carefully.
- To get the recitation code click on the LINK. This will follow to every recitation (just replace the 01 with the number you want in the URL, 02, 03, 04, etc..)
- Practice weekly material with coding problem-sets.

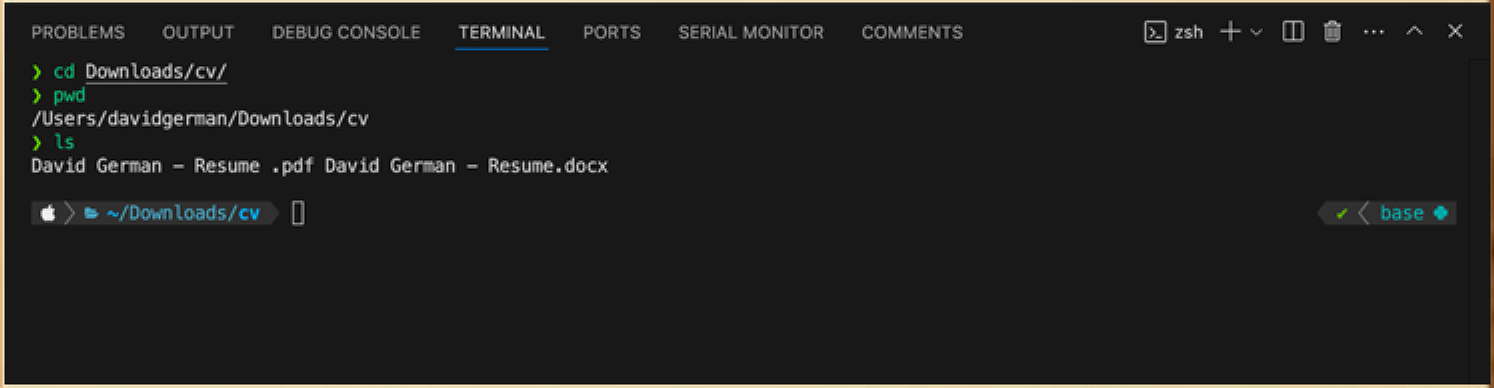
Overview

- Command line
- Git & Github
- Errors
- Variables & Types of variables
- Command line arguments
- Casting
- Math library

Introduction to Computer Science
Reichman University

Recitation 1

Command line / Terminal



```
❯ cd Downloads/cv/
❯ pwd
/Users/davidgerman/Downloads/cv
❯ ls
David German - Resume .pdf David German - Resume.docx

 ~/Downloads/cv                                                  ✔ base
```

Command line - Commands (Java related)

- "javac" is a command which receives an input a java file (file with .java extension and must be specified both name and extension) , the command compiles that java file and creates a .class file.
  - Assume I want to compile the file: MyCode.java
    - "javac MyCode.java"

- "java" is a command which receives an input a compiled java class (class file which shouldn't be specified) and runs that class.
  - Assume I want to run the class: MyCode
    - "java MyCode"
  - Note: it must have a "main" to run
  - If there are usages of args in code they must come after the name of the class in order with a single space between them.

Recitation 1

Git & Github

Git & Github

- Git
  - This is a software that helps developers keep track on their codebase (It has more purposes but for now that what you need to know).
  - Key Terms:
    - Repository - (or Repo foe short) is a project or folder Git watches.
    - Commit – Commit is the current saved version.

- GitHub
  - This is a website where people store their Git-tracked projects so others can see them.
  - It provides a visual interface and hosting for Git repositories.
  - Key Terms:
    - Repository (same as Git, but on GitHub's website),
    - Fork\clone (copying someone else's project to your account).

- Imagine Git as a diary (codebase) you write in every day (commit), you note what you did. Now, if you wanted to share your diary and turn it into a blog with the world or collaborate with others, you'd put it in a library/online (GitHub). If someone likes your story, they borrow it (fork\clone).

Git - Commands

- **git init**
  - What it does: Initializes a new Git repository and starts tracking an existing directory. It adds a hidden subfolder within the existing directory that houses the internal data structure required for version control.
  - Example:
    - (Navigate to the desired directory using the terminal or command prompt).
    - git init.
      - This will convert that directory into a Git repository.

- **git clone**
  - What it does: Copies a Git repository from a remote source (like GitHub) to your local computer.
  - Example:
    - git clone https://github.com/user/repo.git -
      - This will create a folder named repo on your local machine with all the repository's files and Git history.

- **git add**
  - What it does: Stages changes you've made to files for a commit. It's like preparing items you want to pack into a box.
  - Example:
    - git add myfile.txt
      - Stages changes in myfile.txt)
    - git add .
      - Stages changes in all files in the current directory and its subdirectories.

Git - Commands

- **<u>git commit</u>**
  - <u>What it does</u>: Saves the staged changes with a descriptive message, creating a new point in your repository's history. Think of it as sealing the box and labeling it with a note about its contents.
  - Example:
    - 'git commit -m "Fixed a bug in myfile.txt" '
      - commits the staged changes with a message indicating a bug was fixed.

- **<u>git push</u>**
  - <u>What it does</u>: Sends your committed changes to a remote repository (like the one on GitHub). It's like shipping your sealed box to a storage facility for everyone to see or access.
  - Example:
    - "git push origin master" (or "git push origin main")
      - pushes your commits to the master (or main version) of the remote repository called origin.

Git ignore

- **<u>Git Ignore (.gitignore)</u>**
  - The .gitignore file is a special file used by Git to determine which files and directories to ignore, before you make a commit.
  - Why is it used?
    - ❑ Preventing sensitive information (like configuration details), from being added to the repository and leaked.
    - ❑ Excluding automatically generated files (like build artifacts, logs, or compiled binaries), which don't need to be the shared space.
    - ❑ Ensuring unwanted files (like OS-specific files or cache files from various tools) from being included to avoid cluttering the repository.

Introduction to Computer Science
Reichman University

Recitation 1

Errors

Errors - identify & handle

- All basic types of errors can be divided to 3 main types of errors:
  - Compiler Errors
  - Runtime Errors
  - Logical Errors

- Note: There are more types of errors but in this stage the probability you will see one of them is low assuming that safe landing was successful

- Tip: Don't get frustrated, sometimes moving a step back is moving 2 steps forward.
- Tip: If you get frustrated, sometimes clearing your mind can give you a new approach to the solution and solve it.

Errors - Compiler Errors

- This error is caused when you are compiling your code. (javac command)
- Those kind of errors are usually minor and mean that somewhere in your code there is something which doesn't match java guidelines.

- In the terms of semantics, inside your file there is an error
  - Syntax error detected by the compiler. **For example**: "reached the ending without parsing" which mean you forgot a "}" somewhere in your code or "';' expected' which means you forgot a ';' after a statement inside your code.
    - Solution: Find what is the error, and where it occurred, then fix it.
    - Note: Sometimes it is specifying the exact location.

- In the terms of logical errors which cause semantic errors, there are multiple options:
  - There is a mismatch between the declared type variable and given value (when automatic casting isn't applied (example: int num = "1";)
    - Solution: Explicit cast or parse the value.
  - Get a value of a variable which doesn't have initial value (int n; int m = n;)
    - Solution: try and always give initial values.
    - Note: there are languages which allow it (JS for example) but Java doesn't.

Note: There are more

Errors - Runtime Errors

- This kind of errors occur once you are running your code. This type of error can appear once you compiled your file successfully and during the executing the program.
- Some of those are also called ‘Exceptions’ (This topic will be discussed later in the course).
- The nature of these errors are more difficult to solve.

Errors - Runtime Error (Command line arguments)

- Command line arguments are entered upon run call. some errors can be similar to the logical errors of in Compiler Errors, and if they were compiled in the code manually it would've been Compiler Error.
  - For example:
    - int s = Integer.parseInt(args[0]);
    - s = s + 1;
    - java Example
      - Would fail due to not passing enough arguments

  - The file compiled successfully but the input given wasn't defined it causes an error if it were:
    - int s ;
    - s = s + 1;
    - java Example
      - Would fail since it is not knowing what is the value of s

  - Solution: First work with literals declared in the code, then after you solved and tested extensively, shift the literals to command line arguments. once you worked carefully with them. then test with command line arguments.

Errors - Runtime Error (Other)

- In the terms of logical errors :
  - Arithmetic error (Example: 1/0).
  - Calling a negative index in string.
    - Note: Some languages allow it like Python. Java doesn't.
  - Calling an index which exceed the length of the string
    - Note: Some languages allow it like JS. Java doesn't.
  - Try to parse a string which represent a non number into a number. (Example : Integer.parseInt("Hello World"))

- Note: There are many more.
- Note: There is one specifically which means you have a logical error.. its name? StackOverFlow.

- The solution for all these Runtime errors are the same, you must go and **debug** your code.
- The error usually points you toward the "chain" of lines it crashed, you will see a lot of things you didn't wrote but usually near the bottom you will find the part of your code, go from there.

Errors - Logical errors

- These kind of errors occur are the most difficult to detect, since your program is compiling successfully and seems to be running smoothly but something is still not right, and that is what you intended to do.

- How to find them? **Test your code**. Try many cases and edge cases (I.E cases with high likelihood to fail). Knowing your code inside and out and be sure that you know what is the expected output at every stage and see that your actual output is matching.

- There are two main types:
  - Your program doesn’t stop running (solution: ctrl+c in windows, command+c in mac).
  - Your program stop running.

- Either way after each you should see what causes the faulty output and change it in a way which doesn’t change the output of previously checked code and solve the new test case.
- Try to generalize the program to solve as many cases as possible without causing errors to original values.

VS Code

VS (Visual Studio) Code

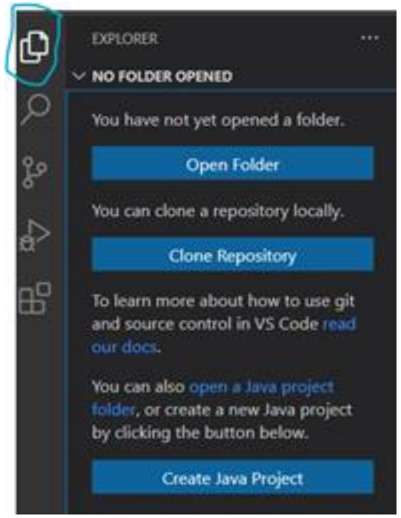
- VS code is a free Integrated Development Environment (IDE) for many programming languages, including JAVA.

- Provides many features to support and ease the programming process such as: Syntax highlighting, and Intelligent code completion.

VS Code – Writing your first program

- The Java programs are written and maintained in an environment called "Java Project" (similar to the notion of a folder)

- Create a new project:
  - Click the EXPLORER icon (marked here)
  - Click "Create Java project"
  - Choose "no build tools"



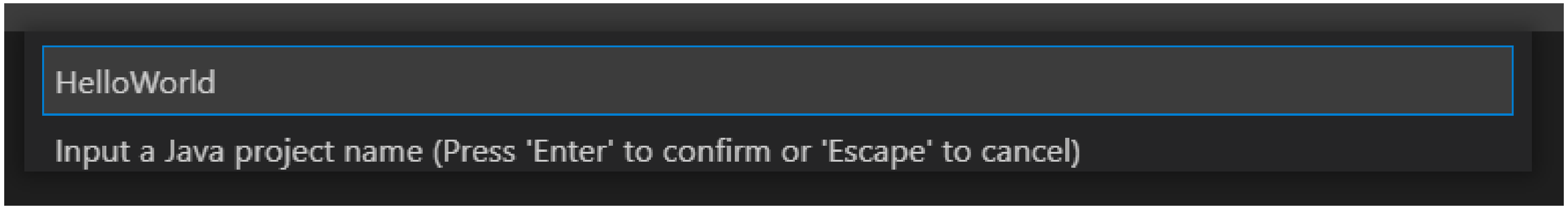
You have not yet opened a folder.

You can clone a repository locally.

To learn more about how to use git and source control in VS Code read our docs.

You can also open a Java project folder, or create a new Java project by clicking the button below.

- Choose your preferred project location (e.g. Intro2CS folder)
- Name your project (e.g. HelloWorld)

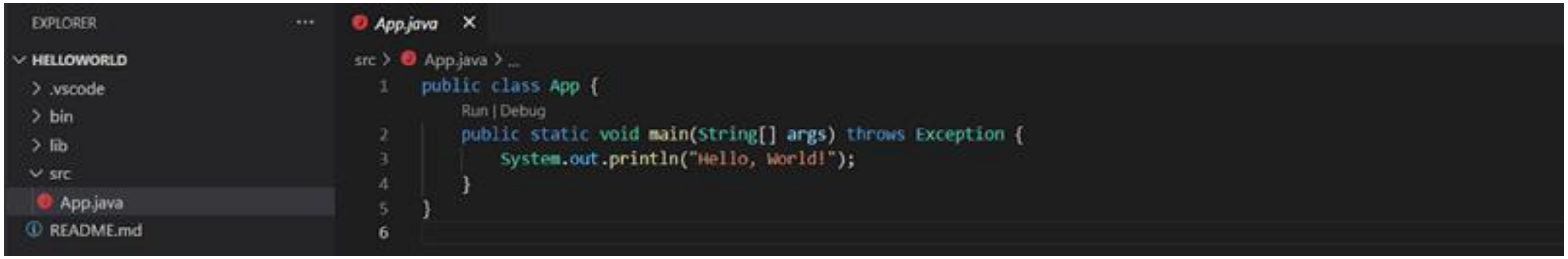

HelloWorld

Input a Java project name (Press 'Enter' to confirm or 'Escape' to cancel)

VS Code – Writing your first program

- Your project will now open

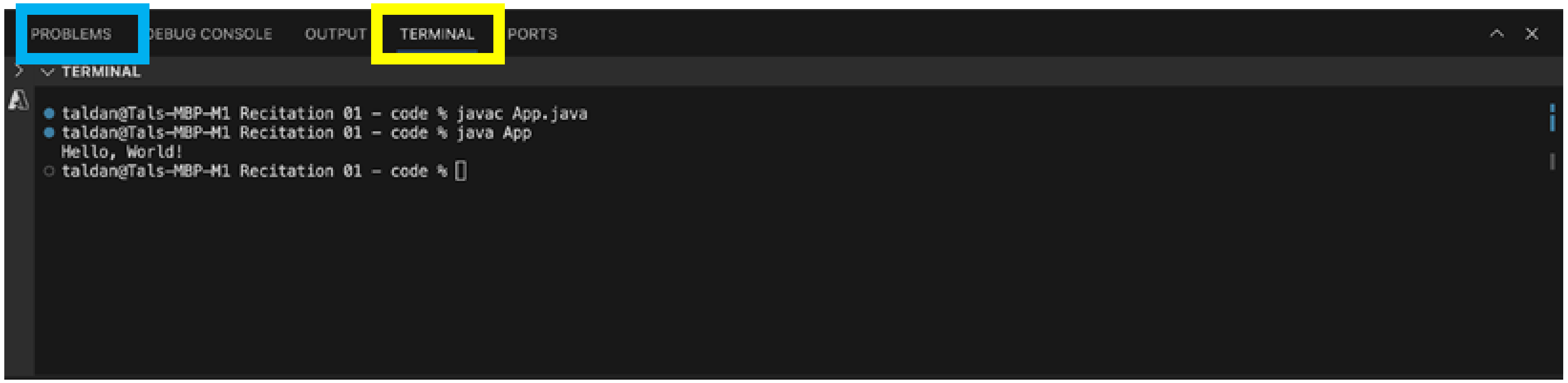
- Your code will be stored under the “src” folder, with a default class called App.java



```java
public class App {
    public static void main(String[] args) throws Exception {
        System.out.println("Hello, World!");
    }
}
```

VS Code – running a program

- This is what you will see for the HelloWorld code from the previous slide:

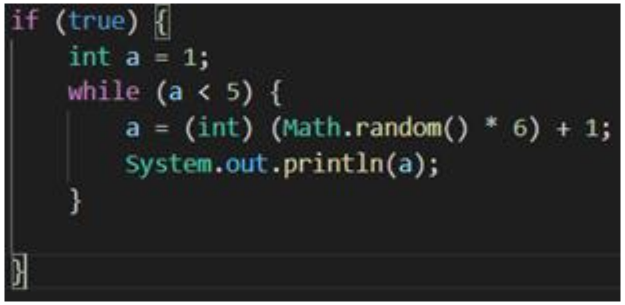


```
● taldan@Tals-MBP-M1 Recitation 01 - code % javac App.java
● taldan@Tals-MBP-M1 Recitation 01 - code % java App
  Hello, World!
○ taldan@Tals-MBP-M1 Recitation 01 - code % 
```

- This TERMINAL window (yellow) is similar to the terminal/cmd shell used before
  - The run button replaces the compilation and execution loop
  - PROBLEMS tab displays compilation and runtime errors (blue)

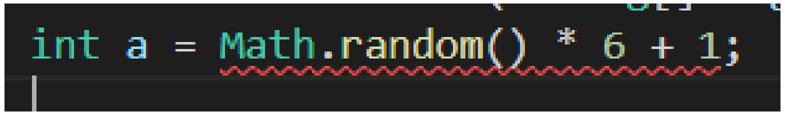
VS Code – Syntax highlighting and error management

- Different types of code elements are marked by different colors:



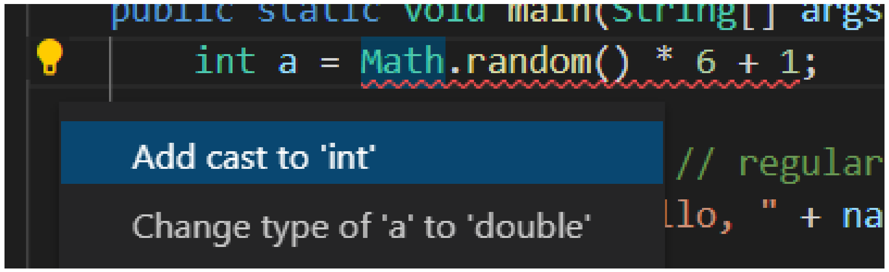
```
if (true) {
    int a = 1;
    while (a < 5) {
        a = (int) (Math.random() * 6) + 1;
        System.out.println(a);
    }

}
```

- Errors are marked by a red line:



```
int a = Math.random() * 6 + 1;
```

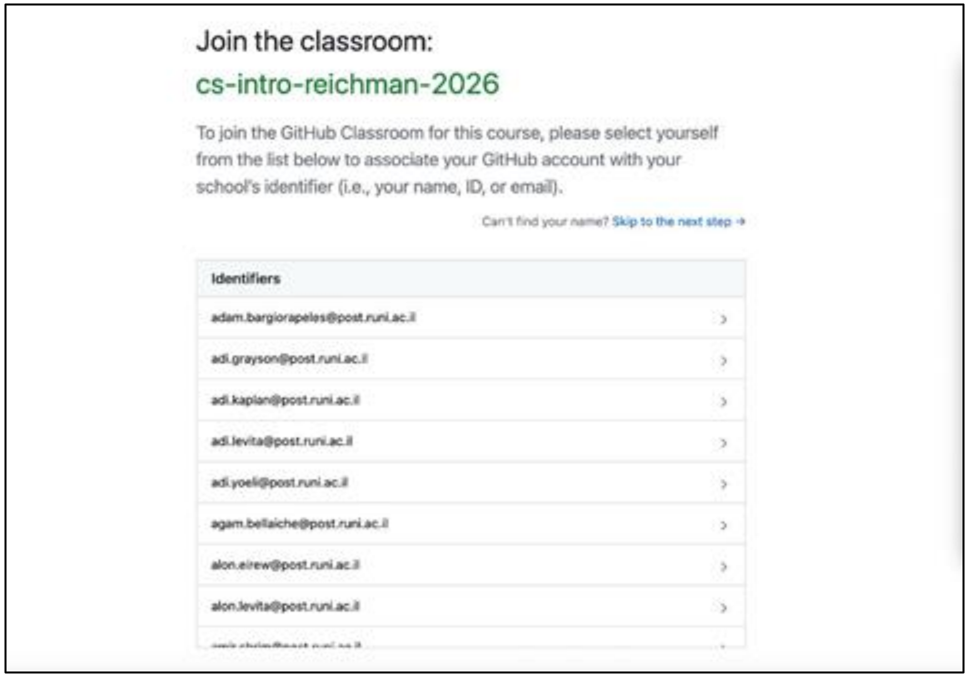
- Hovering the mouse over the text will show an explanation for the error
- Pressing the lightbulb at the beginning of the line (if it exists) will offer a tip to solve the error



```
public static void main(String[] args
    int a = Math.random() * 6 + 1;
```

HW Submission

- Guide on how to start your HWs is on Moodle, and ensure you are paired to your university email, <u>if you have an issue please contact us, don't skip this step.</u>



Join the classroom:

cs-intro-reichman-2026

To join the GitHub Classroom for this course, please select yourself from the list below to associate your GitHub account with your school's identifier (i.e., your name, ID, or email).

Can't find your name? Skip to the next step →

Identifiers
adam.bargiorapeles@post.runi.ac.il
adi.grayson@post.runi.ac.il
adi.kaplan@post.runi.ac.il
adi.levita@post.runi.ac.il
adi.yoeli@post.runi.ac.il
agam.bellaiche@post.runi.ac.il
alon.eirew@post.runi.ac.il
alon.levita@post.runi.ac.il
[illegible]

Introduction to Computer Science
Reichman University

Recitation 1

Variables & Types of Variables

Question 1: Flip Flop

- Write a simple Java program that swaps the values of two integer variables without using any operations or losing data.
- Example:
  - If a=5 and b=7 were the values in the start of the program, by the end of the program the values a=7, b=5.

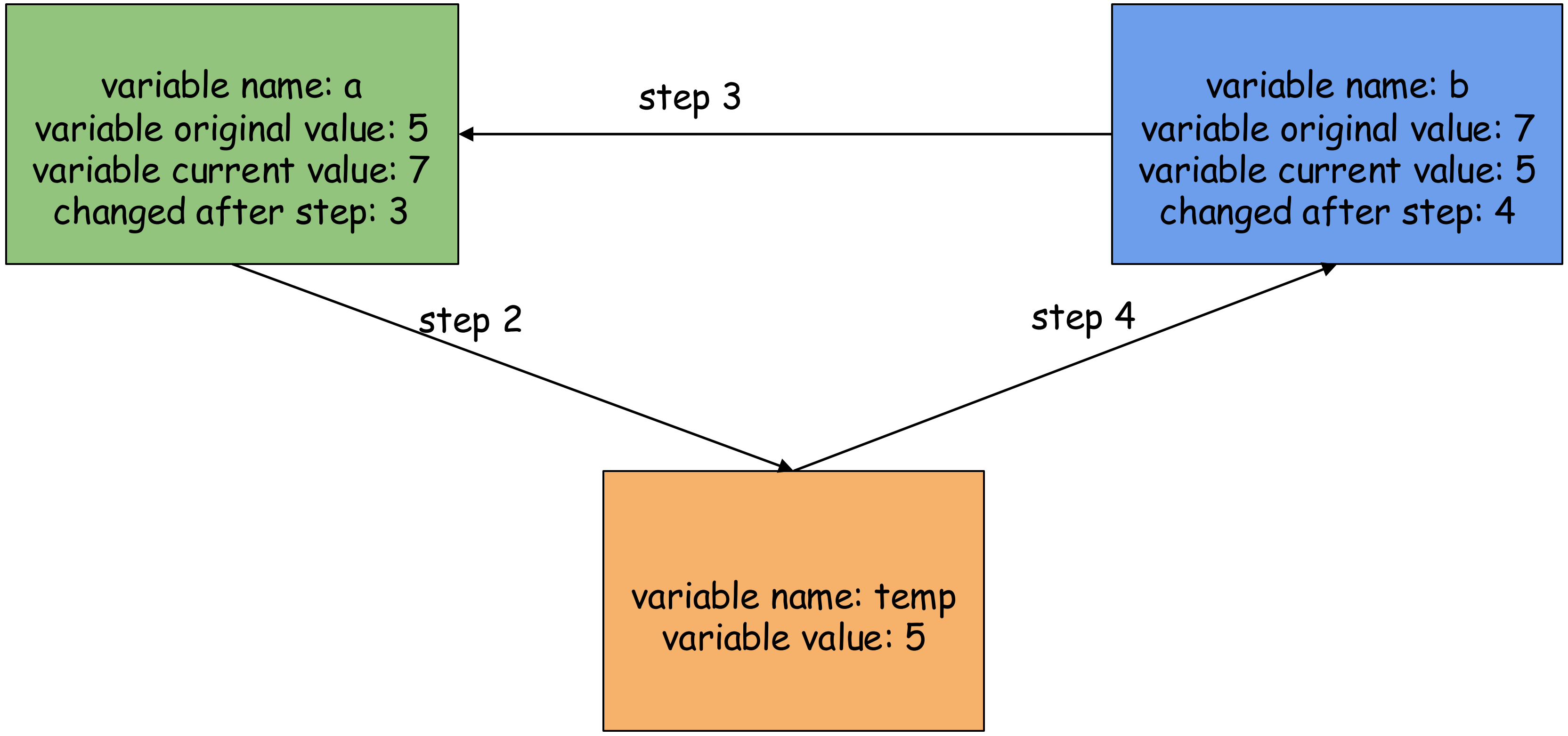
Question 1: FlipFlop - Strategy Illustration

Step 1: Initial declaration of 'a' and 'b'.

Step 2: Declaration of 'temp' and give it the initial value stored in 'a'.

Step 3: Reassign the value of the variable 'a' to be the value store in variable 'b'.

Step 4: Reassign the value of the variable 'b' to be a copy of 'a' which is stored in 'temp'.



Question 1: Flip Flop - Solution

```
/*
* This program initializes two integers with arbitrary
* values then flips their values. Meaning, the first
* variable will now have the value of the second variable
* and vice versa.
*/
public class FlipFlop {
   public static void main(String[] args){
     int a = 5;
     int b = 7;
     int temp = a;
     a = b;
     b = temp;
   }
}
```

Question 1, Expansion 1: Flip Flop

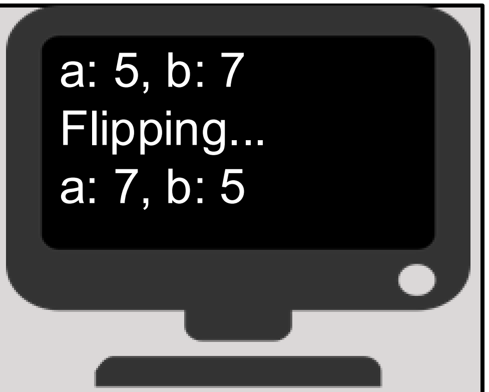
- We are going to add System.out.println() lines.
- In programming it is a very common practice to use printing lines, they have multiple usages among developer, here are few examples:
  - **Debugging**: One of the primary reasons programmers use print statements is to debug code. Printing the values of variables, flow control indicators, or other significant states can help you trace the execution of a program and identify where a logic error might be occurring.
  - **Documentation and Learning**: For beginners, printing out results and intermediary steps can be extremely beneficial in understanding how a piece of code or an algorithm works.
  - **Validation of Assumptions**: Developers often make assumptions about the state of the program, the nature of data, or the outcome of a calculation. It's often useful to print it out to ensure it meets expectations.

Question 1, Expansion 1: Flip Flop - Solution

```java
// Initializes two arbitrary variable, flips them
// and prints the process
public class FlipFlop {
   public static void main(String[] args){
     int a = 5;
     int b = 7;
     System.out.println("a: " + a + ", b: " + b);         // added line
     System.out.println("Flipping…");                     // added line
     int temp = a;
     a = b;
     b = temp;
     System.out.println("a: " + a + ", b: " + b);         // added line
   }
}
```

Question 1, Expansion 1: Flip Flop - Solution

```java
// Initializes two arbitrary variable, flips them
// and prints the process
public class FlipFlop {
   public static void main(String[] args){
     int a = 5;
     int b = 7;
     System.out.println("a: " + a + ", b: " + b);          // added line
     System.out.println("Flipping…");                      // added line
     int temp = a;
     a = b;
     b = temp;
     System.out.println("a: " + a + ", b: " + b);          // added line
   }
}
```



```
a: 5, b: 7
Flipping...
a: 7, b: 5
```

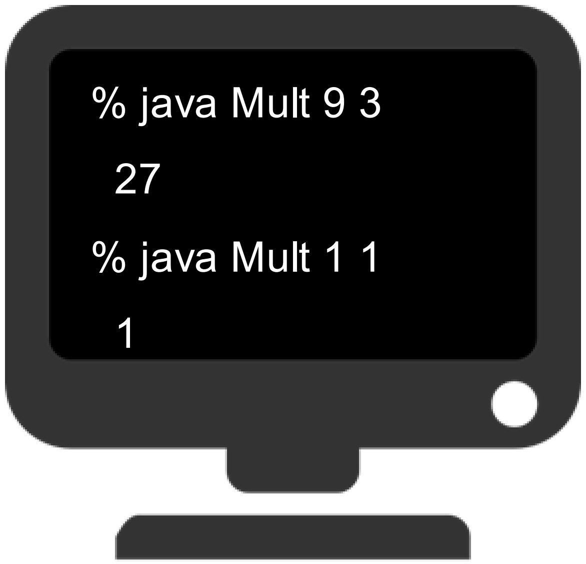
Introduction to Computer Science
Reichman University

Recitation 1

Command Line Arguments

Command Line Arguments - What are they?

- A Java program can accept any number of arguments from the command line.
- This allows the user to specify variable values before starting a program.
- The user enters command-line arguments when running the program and specifies them after the name of the class to be run.
- **For example**: suppose a Java program named *Mult* reads two command-line integers from the users, multiplies them and prints the solution.
- A typical call to *Mult* will look like this:



```
% java Mult 9 3
 27
% java Mult 1 1
 1
```

Question 1, Expansion 2: Flip Flop - command line arguments

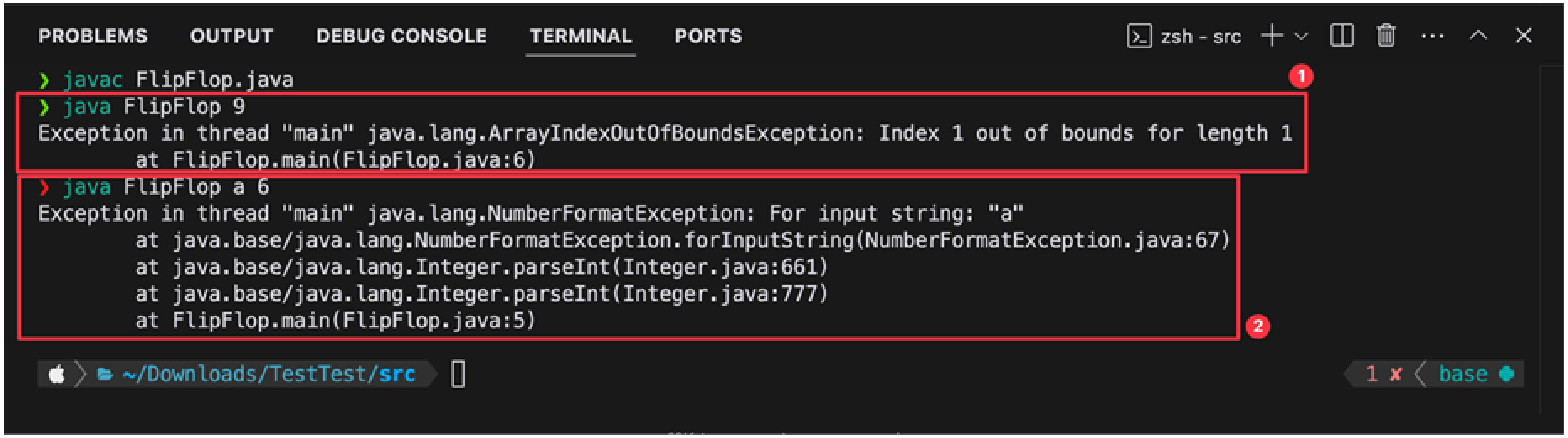
- We are going to add adjust the program to receive command line arguments. In Java programming it is common practice to use command line arguments. using command line argument has multiple advantages among them.
- **Flexibility:** Command line arguments make a program more versatile. Instead of hardcoding certain values, you can specify them at runtime. This means that you can use the same program for different inputs without modifying the code.
- **Testing & Debugging**: While developing, you can use command line arguments to test specific code paths or scenarios without compiling your code again. This can be especially handy when debugging.
- **Common Convention**: Many command-line tools and utilities use command line arguments. If you're developing a tool intended for command line usage, utilizing arguments makes your tool consistent with familiar command line conventions.

Using Command Line Arguments

- Command-line arguments can be accessed using args[i] inside **main**. while replacing i with the place in line of the desired argument, starting from 0.
- So, in "**% java Mult 9 3**", args[0] is 9 and args[1] is 3.
- Every such argument is a **String** until 'proven' otherwise.
- The process of allocating an argument to the correct variable type is called parsing.
- Every type has a unique parsing technique.
  - 'int' – Integer.parseInt(args[0]);
  - 'double' – Double.parseDouble(args[0]);
  - 'boolean' – Boolean.parseBoolean(args[0]);

Using Command Line Arguments - Warnings

- Two things to look out for:
  - The user needs to give as many arguments as the program expects to get.
  - The arguments' type need to match the types the program expects to receive.
- Failing to meet the program's 'expectations' can result in errors.



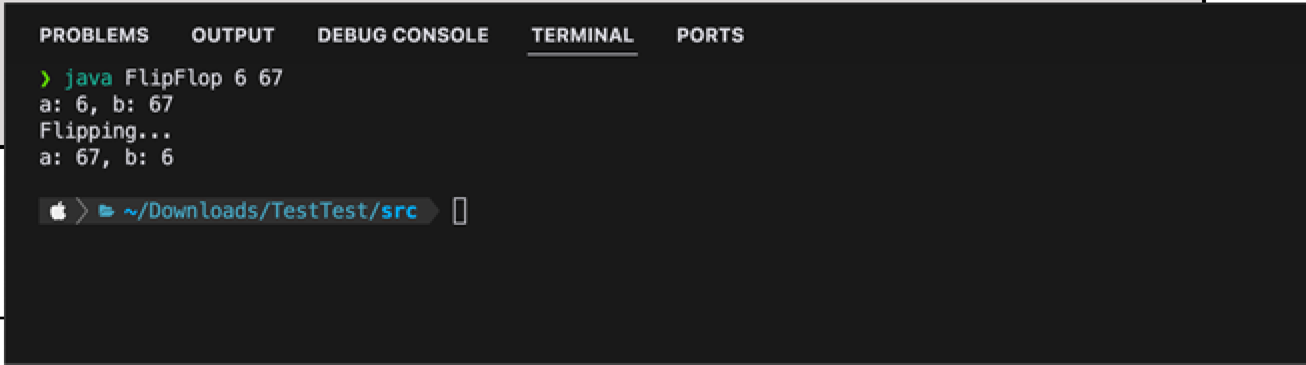
```
❯ javac FlipFlop.java
❯ java FlipFlop 9
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: Index 1 out of bounds for length 1
        at FlipFlop.main(FlipFlop.java:6)
❯ java FlipFlop a 6
Exception in thread "main" java.lang.NumberFormatException: For input string: "a"
        at java.base/java.lang.NumberFormatException.forInputString(NumberFormatException.java:67)
        at java.base/java.lang.Integer.parseInt(Integer.java:661)
        at java.base/java.lang.Integer.parseInt(Integer.java:777)
        at FlipFlop.main(FlipFlop.java:5)
 ~/Downloads/TestTest/src                                            1 ✘  base
```

Question 1, Expansion 2: Flip Flop - Solution

```java
// Reads two integers from the user, flips them
// and prints the process

public class FlipFlop {

   public static void main(String[] args){

     int a = Integer.parseInt(args[0]); //adjusted line

     int b = Integer.parseInt(args[1]); //adjusted line

     System.out.println("a: " + a + ", b: " + b);

     System.out.println("Flipping…");

     int temp = a;

     a = b;

     b = temp;

     System.out.println("a: " + a + ", b: " + b);

   }

}
```

Question 1, Expansion 2: Flip Flop - Solution

```java
// Reads two integers from the user, flips them
// and prints the process

public class FlipFlop {

   public static void main(String[] args){

     int a = Integer.parseInt(args[0]); //adjusted line

     int b = Integer.parseInt(args[1]); //adjusted line

     System.out.println("a: " + a + ", b: " + b);

     System.out.println("Flipping…");

     int temp = a;

     a = b;

     b = temp;

     System.out.println("a: " + a + ", b: " + b);

   }

}
```



```
❯ java FlipFlop 6 67
a: 6, b: 67
Flipping...
a: 67, b: 6

 ~/Downloads/TestTest/src
```

Using Command Line Arguments - Question

- Consider the following code:

```
public class Mystery {
   public static void main(String[] args){
     int a = Integer.parseInt(args[0]);
     int b = Integer.parseInt(args[2]);
     …
   }
}
```

- Assume there are no more other args calls in the program.
- How many arguments does the program expect and why?
  - **Answer**: 3
  - **Reason**: It expects 2 arguments in the 0 and 2 spots, so it expects another argument (in position 1) but it never used. So a second argument is expected but never used..

Recitation 1

Type Conversion - Basics

Type Conversion – Examples

- `int i = 2 / 4 ;` `// i is 0`
- `double d = 2 / 4;` `// d is 0.0`
- `double d = 2 / 4.0;` `// d is 0.5`
- `int i = (double) 2 / 4 ;` `//cannot convert`
- `double d = ((double)2) / 4;` `//d is 0.5`
- `double d = (double) 2 / 4;` `//d is 0.5`
- `double d = (double) (2 / 4);` `//d is 0.0`

Question 2 – Type Conversion

- As we know when we cast from double to int the value always rounded down, in some cases it would be wise to round to the nearest integer.
- **I.E**: If the floating-point reminder of the number is bigger or equal than 0.5, the number will be rounded up to the next integer, otherwise, the number will be rounded down to previous integer.
- **Constraint**: For the following question **you may not allow** to use Math.round(int a) and conditions.
- Design a program which does the following:
  - The program receives a floating number from the user.
  - Then, the program preform the rounding operation (mentioned above)
  - Lastly, it prints the original value and the result of the operation.

Question 2 - Solution

```
public class Rounding {
    public static void main(String[] args) {
        double value = Double.parseDouble(args[0]);
        int rounded = (int) (value + 0.5);
        System.out.println("Rounded value of " + value + " is: " + rounded);
    }
}
```

Introduction to Computer Science
Reichman University

Recitation 1

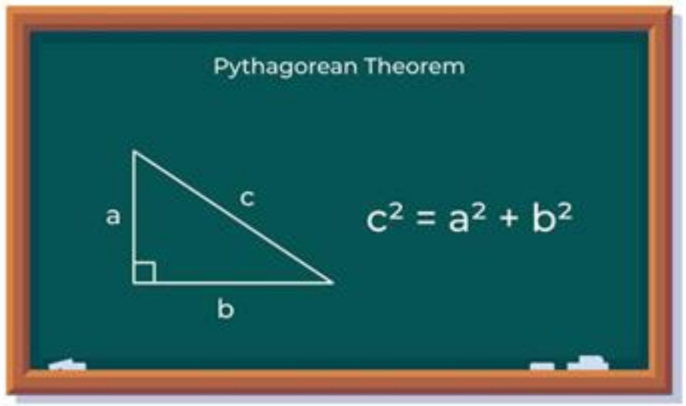
Math Library

Math

- The Math library supports a great variety of mathematical operations.
- For complete information and documentation visit the Java Math API at http://docs.oracle.com/javase/8/docs/api/java/lang/Math.html

Question 3 – Math

- The Pythagorean theorem is a fundamental result in geometry that applies to right-angled triangles. In its simplest form it says:
  - Take any triangle with one angle exactly 90° (a "right triangle"). Label the two legs that meet at the right angle as a and b, and the side opposite the right angle (the longest side) as c.
- Design a program which does the following:
  - Receives a couple of integers (int values) from the user.
    - ❑ You may assume that the inputs are positive valid numbers.
  - Calculates the length of the hypotenuse in a right triangle assuming both integers are the other sides.
  - Returns the calculated hypotenuse value.
  - Reminder: a^2 + b^2 = c^2



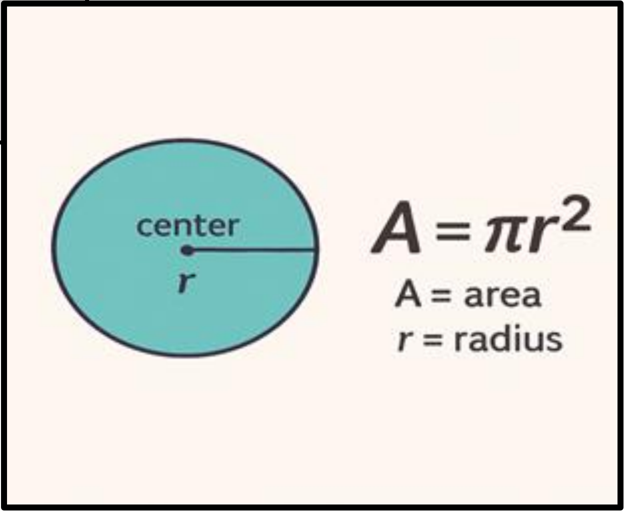
Question 3 - Solution

```
public class PythagoreanTheorem {
        public static void main(String[] args){
            // Getting the data from command line arguments
            int a = Integer.parseInt(args[0]);
            int b = Integer.parseInt(args[1]);

            // Raising the values to the power of 2 and convert them into int
            int aSquared = (int) Math.pow(a, 2);
            int bSquared = (int) Math.pow(b, 2);

            // Adding them up and square rooting
            int cSquared = aSquared + bSquared;
            double c = Math.sqrt(cSquared);
            System.out.println(a + "^2 + " + b + "^2 = " + c + "^2");
        }
}
```

Question 4 – Math

- Design a program which does the following:
  - Receives a single int from the user.
    - ❑ If negative turn into positive (don't need if statement)
    - ❑ The value represent the radius of some circle (π * radius^2)
  - The program prints the area of a circle with the radius.

```
public class CircleArea {
        public static void main(String[] args){
                // Your code here
        }
}
```



Question 4 - Solution

```
public class CircleArea {
        public static void main(String[] args){
            int input = Integer.parseInt(args[0]);
            int radius = Math.abs(input);

            double area = Math.PI * Math.pow(radius, 2);
            System.out.println("Radius: " + radius + " , Area: " + area);
        }
}
```

Question 5 - Dice Roll

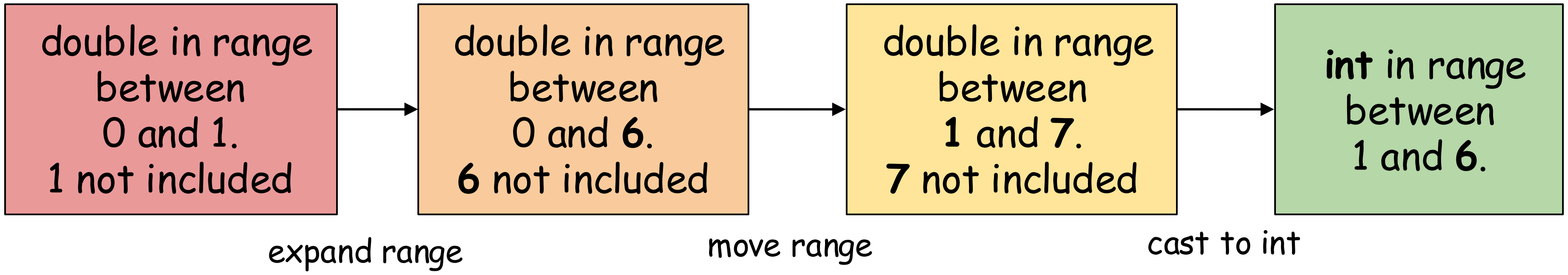
- Many board games have an element of dice roll. a fair 6-sided dice have the numbers between 1 and 6 (included).
- Design a Java program that simulates the roll of a six-sided dice.
  - when executed, the program should generate a random number between 1 and 6 (inclusive) to represent the outcome of a dice roll. Display the result to the user.
  - Ensure that each number between 1 and 6 has an equal chance (as possible) of being rolled (AKA fair 6-sided dice). You may use Math library freely in the solution.
- That sounds like a complicated problem, how to approach the problem?

Question 5 - Strategy

- First, identify what can be used.
  - We can use Math.random(), to generate a random number.
- Second, identify if the problem is solved, and if not, what are the problems we need to tackle solve?
  - No, the function Math.random(), returns a <u>**double**</u> between <u>**0 and 1(not included)**.</u> So, we have 2 problems:
    - The range of the roll is not correct (which breaks into 3 smaller problems)
      - Minimum value of range is not correct (0 instead of 1)
      - Maximum value of range is not correct (1 instead of 6)
      - Maximum value of range is not included (should be included)
    - The type of value (double instead of int)

Question 5 - Strategy

- Now we solve those problems:
  - For both maximum problems: We can multiply the result by the difference of range + 1. Example: 0.5 * 6 = 3, 0.66667 * 6 = 4 etc..
    - **Expand the range**
  - For the minimum value: We can add 1 to the result. If rolled a 0.1 which will turn to 0, after adding 1.
    - **Move the range**
  - As for the type of value and the maximum value not included: We can cast the double into int.
    - **Cast to int**
- Last, lets combine the solutions in the right order:


double in range between 0 and 1. 1 not included
double in range between 0 and 6. 6 not included
double in range between 1 and 7. 7 not included
int in range between 1 and 6.
expand range
move range
cast to int

Question 5 - Solution - step by step

- **Print a real number between 0 and 1 (Phase 1)**

```
System.out.println(Math.random())
```

- **Print a real number between 0 and 6 (Phase 2)**

```
System.out.println(Math.random() * 6)
```

- **Print a real number between 1 and 7 (Phase 3)**

```
System.out.println(Math.random() * 6) + 1
```

- **Print an integer between 1 and 6 (Phase 4)**

```
System.out.println((int)((Math.random() * 6) + 1))
```

Question 5 - Solution

```
public class DiceRoll {
        public static void main(String[] args){
            System.out.println((int) ((Math.random() * 6) + 1));
        }
}
```

Question 5, Expansion 1 – Non-6-Sided DiceRoll

- Some games have other 6 options for a randomness:
  - If I were to shuffle a deck of cards, I have 52 options to draw a given card. (13 cards for each suit).
  - In D&D, there are dice with various number of sides (for example: 4-sided dice, 8-sided dice, 10-sided dice, 20-sided dice etc.).
- Design a Java program that simulates the roll of an N-sided dice.
  - When executed, the program receives a positive int named "max" which represent N (such that N > 1) from the user.
  - Then the program should generate a random number between 1 and N (inclusive) to represent the outcome of a dice roll. Display the result to the user.
  - Ensure that each number between 1 and N has an equal chance (as possible) of being rolled (AKA fair N-sided dice). You may use Math library freely in the solution.

Question 5, Expansion 1 - Solution

```
 public class DiceRoll {
         public static void main(String[] args){
                   int max = Integer.parseInt(args[0]);
                   System.out.println((int) ((Math.random() * max) + 1));
         }
}
```

Question 5, Expansion 2 - Random range

- Some games have other may include negative numbers or numbers which are strictly bigger than 1.
  - If I was to ask someone to pick a number between -10 and 10.
  - If I was to ask someone to pick a number between 4 and 16.
- Design a Java program that simulates the pick of an int number in range [M,N]. (both M and N are included)
  - When executed, the program receives from the user, a couple of int values named "max" and "min" which represent N,M (respectively).
  - Then the program should generate a random number between M and N (inclusive) to represent the outcome of a number pick. Display the result to the user.
  - Ensure that each number between M and N has an equal chance (as possible) of being rolled (AKA fair odds). You may use Math library freely in the solution.

Question 5, Expansion 2 - Solution

```
public class DiceRoll {
        public static void main(String[] args){
          int max = Integer.parseInt(args[0]);
          int min = Integer.parseInt(args[1]);
          System.out.println((int) ((Math.random() * (max - min + 1)) + min));
        }
}
```

- **Generalize the random in range formula (both work)**

    `(int)(Math.random() * (max - min + 1)) + min`

    `(int)(Math.random() * (max - min + 1) + min)`