Introduction to Computer Science
Reichman University

Recitation 2

In Memoriam

- This recitation is dedicated to the memory of Sargent Major (res') Joseph (Joe) Gitarts. Joe fell on December 25th 2023 during a battle in South Gaza. Joe was scheduled to start his second year here at RUNi when he fell. May he rest in peace.

Today

- Strings methods
- Flow Control
  - Conditions - if, else, else if, switch
  - Loops - while, for
  - Shorthand operations
- Variable Scope

Question 1 - Warmup

- The Euclidean distance is the straight-line distance between two points in a coordinate system.
- For two points $P_1(x_1, y_1)$ and $P_2(x_2, y_2)$ in 2D space, it is defined as:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

- Write a simple Java program that receives the 2 points as 4 command line arguments (x1, y1, x2, y2) and prints the distance between those points.
- **Constraint:** You may only use the Math.sqrt and Math.pow function as well as the parsing functions.

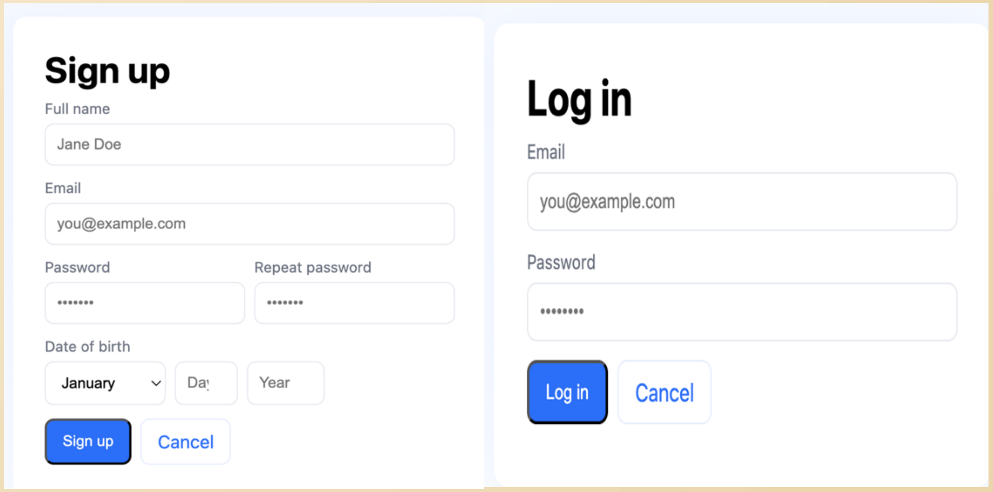
Question 1

```
public class Distance2D {

    public static void main(String[] args) {

        // Parse command-line arguments

        double x1 = Double.parseDouble(args[0]);

        double y1 = Double.parseDouble(args[1]);

        double x2 = Double.parseDouble(args[2]);

        double y2 = Double.parseDouble(args[3]);


        // Compute squared differences

        double dxSquared = Math.pow(x2 - x1, 2);

        double dySquared = Math.pow(y2 - y1, 2);


        // Compute Euclidean distance

        double distance = Math.sqrt(dxSquared + dySquared);


        // Print result

        System.out.println("The distance between the two points is: " + distance);

    }

}
```

Introduction to Computer Science
Reichman University

Recitation 2

Signup & Login



Story for Today

- In many websites today, there is a login and signup pages, those pages are integral to
- **Sign-Up Page (Registration)**
  - Collects **user details**: name, email, password, and date of birth (and more).
  - Used for **creating a new account** in the system.
- **Log-In Page (Authentication)**
  - Used for **existing users to access** their accounts.
  - Requires **email + password**.
  - Specific users might have different permissions to certain applications or inside the application, and this permission fetching is usually connected to the login function.
- In Today's recitation we are going to build a simplified version of those pages.

Recitation 2

Strings

string str = 'Hello World';
I PREFER THE MOST FAMOUS STRING IN ITS RIGHT FORM
string str = " Hello World ";
I SAID THE RIGHT FORM
String str = " Hello World ";
PERFECTION

Strings - Examples

```
public class StringExample {
            public static void main(String[] args){
               String str = "Hello";
               System.out.println(str.length()); // Prints the length of str 5.
               System.out.println(str.charAt(1)); // Prints the second char ‘e’.
               System.out.println(str.charAt(7)); // Try to print non existing char.
               System.out.println(str.charAt(-1)); // Try to print non existing char.
               System.out.println(str.indexOf(‘o’)); // Prints 1st appearance of ‘o’. 4
               System.out.println(str.indexOf(‘l’)); // Prints 1st appearance of ’l’. 2
               System.out.println(str.indexOf(‘m’)); /* Prints 1st appearance of ‘m’.didn’t find first; -1. */
               System.out.println(str.indexOf(‘h’)); /* Prints 1st appearance of ‘h’ didn’t find (case sensitive); -1.*/
               System.out.println(str.substing(2)); /*  Prints the string from index 2 to end; "llo"*/
               System.out.println(str.substing(1, 3)); /*  Prints the string from index 1 to index 3 (not included)
                                                                ; "el"*/


            }
}
```

Question 2: Is Valid Email

- Any email share one common trait. Each email has exactly one '@', followed by a suffix. For now we will write a simple program which checks if there is '@' in a given string.
- Write a simple Java program that receives a command line argument, and prints if there is a '@' in the string.

Question 2 - Solution

```
public class IsVaildEmail {
        public static void main(String[] args){
           String str = args[0];
           int index = str.indexOf(‘@’);
           boolean isValid = index != -1;
           System.out.println("Email validity: " + isValid);
        }
}
```

Introduction to Computer Science
Reichman University

Recitation 2

Control Flow - Conditions

Question 2, Expansion 1: Is Valid Email

- We will now upgrade the solution to include if there is a single '@' in the given string.
- Write a simple Java program that receives a command line argument, and prints if there is a single '@' in the string.

Question 2, Expansion 1 - Solution

```
public class IsVaildEmail {
            public static void main(String[] args){
               String str = args[0];
               int index = str.indexOf(‘@’);
               boolean hasAt = index != -1;
               if (!hasAt) {
                 // There is no ‘@’ in string
                 System.out.println("String has no ‘@’");
               } else {
                 String rem = str.substring(index);
                 int index2 = rem.indexOf(‘@’);
                 if (index2 == -1) {
                    // There is no ‘@’ in reminder
                    System.out.println("String has is possible email");
                 } else {
                    // There is ‘@’ in reminder
                    System.out.println("String has multiple ‘@’");
                 }
            }
}
```

Question 2, Expansion 2: Is Valid Email

- We will now upgrade the solution to include if there is a single '@' in the given email and we know that after the '@' there is a suffix. Hence it shouldn't be the last char of the string.
- Write a simple Java program that receives a command line argument, and prints if there is a single '@' in the string.

Question 2, Expansion 2 - Solution

```
public class IsVaildEmail {
           public static void main(String[] args){
              String str = args[0];
              int index = str.indexOf('@');
              boolean hasAt = index != -1;
              if (!hasAt) {
                // There is no '@' in string
                System.out.println("String has no '@'");
              } else {
                String rem = str.substring(index);
                int index2 = rem.indexOf('@');
                if (index2 == -1 && ((str.length() – 1) != index ) {
                   // There is no '@' in reminder, AND index is not the last index of the last char
                   System.out.println("String has is possible email");
                } else {
                   // There is '@' in reminder
                   System.out.println("String has multiple '@'");
                }
           }
}
```

Question 3: Assign user role

- As we mentioned on login the user get a role.
- Assume that in our system there are 3 main roles:
  - Customer – Sees the least data & functionalities.
  - Manager – Sees more data & functionalities.
  - Admin - Sees all data & functionalities.
- In simple systems those roles are assigned numbers.
- In our system let's assume that:
  - Customer = 1.
  - Manager = 2.
  - Admin = 3.
- Write a simple Java program that given a number (as command line argument) will print the user's role (as String)

Question 3 - Solution

```
public class UserRole {
   public static void main(String[] args){
     int roleNumber = Integer.parseInt(args[0]);
     String role = "";
     if (roleNumber >= 3){
         role = "Admin";
     } else if (roleNumber >= 2){
         role = "Manager";
     } else
         role = "Customer";
     }
     System.out.println("Users role is: " + role);
   }
}
```

Question 3, Expansion 1 - Solution

- Alice solved the code as follows:

```
public class UserRole {
   public static void main(String[] args){
     int roleNumber = Integer.parseInt(args[0]);
     String role = "";
     if (roleNumber >= 2){
         role = "Manager";
     } else if (roleNumber >= 3){
         role = "Admin";
     } else
         role = "Customer";
     }
     System.out.println("Users role is:  " + role);
   }
}
```

- Will Alice get the same result?

Question 3, Expansion 1 - Solution

- Alice solved the code as follows:

```
public class UserRole {
   public static void main(String[] args){
     int roleNumber = Integer.parseInt(args[0]);
     String role = "";
     if (roleNumber >= 2){
         role = "Manager";
     } else if (roleNumber >= 3){
         role = "Admin";
     } else
         role = "Customer";
     }
     System.out.println("Users role is: " + role);
   }
}
```

- Will Alice get the same result?

- No!
- Else if segment is never reached.
- Admin will receive "Manager".

Question 3, Expansion 2 - Solution

- Bob solved the code as follows:

```
public class UserRole {
   public static void main(String[] args){
      int roleNumber = Integer.parseInt(args[0]);
      String role = "";
      if (roleNumber >= 3){
          role = "Admin";
      }
      if (roleNumber >= 2){
          role = "Manager";
      } else {
          role = "Customer";
      }
      System.out.println("Users role is: " + role);
   }
}
```

- Will Bob get the same result?

Question 3, Expansion 2 - Solution

- Bob solved the code as follows:

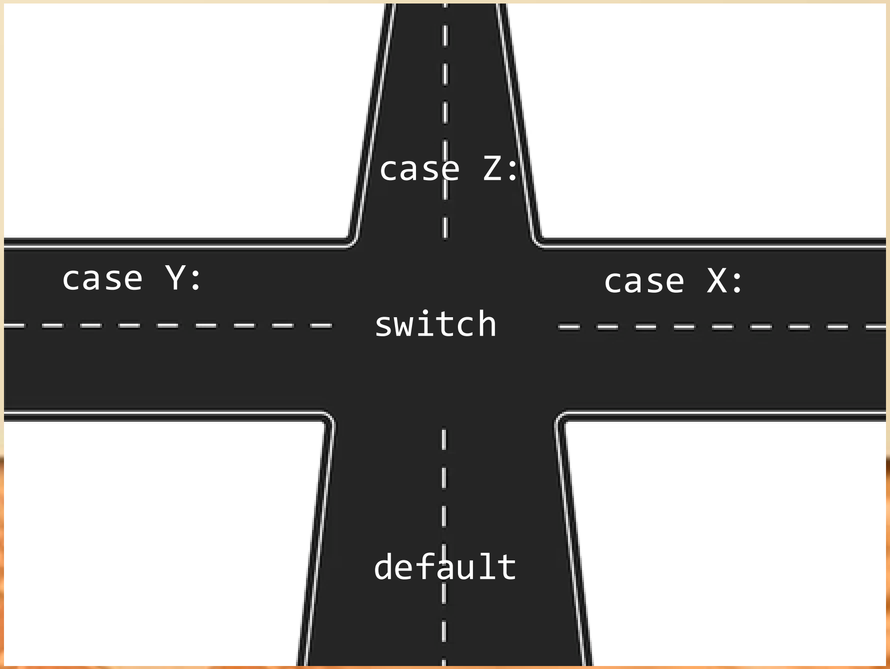
```
public class UserRole {
   public static void main(String[] args){
     int roleNumber = Integer.parseInt(args[0]);
     String role = "";
     if (roleNumber >= 3){
         role = "Admin";
     }
     if (roleNumber >= 2){
         role = "Manager";
     } else {
         role = "Customer";
     }
     System.out.println("Users role is: " + role);
   }
}
```

- Will Bob get the same result?

- No!
- Else if segment is never reached.
- Admin will receive "Manager".

Recitation 2

Control Flow - Switch



The Switch Statement

```
switch variable {
    case value1:
        // code segment 1

        break;
    case value2:
        // code segment 2

        break;
   ...
    default:
       // default code segment

        break;

}
```

- The expression value is compared to each case value; if there's a match, the corresponding statement is executed.
- If none of the values matches, the default statement is executed (if exist).
- **default** is optional and equivalent to **else**.
- The type of the variable and the values must be integral (**byte**, **short**, **char**, **int**), or **String**.
- The flow of control continues through all the cases, even after a match.
- To get out in each case (including default), use **break**.

Question 4 – Switch Case

- When we use the Gregorian calendar, each month is assigned a number (1 -> “January”, etc.)
- Design a program which translates from number to Month name.
  - Receives a number of month from the user
  - If it is a valid month number, (IE in the the range of 1-12), it returns the name of the respective month (1 -> “January”, etc.), otherwise, put an “Invalid month”.
  - Then prints the solution.

Question 3 - Solution

```
int month = Integer.parseInt(args[0]);
String monthString = "";
switch (month) {
    case 1:  monthString = "January";
             break;
    case 2:  monthString = "February";
             break;
    case 3:  monthString = "March";
             break;
    case 4:  monthString = "April";
             break;
    case 5:  monthString = "May";
             break;
    case 6:  monthString = "June";
             break;
    case 7:  monthString = "July";
             break;
    case 8:  monthString = "August";
             break;
    case 9:  monthString = "September";
             break;
    case 10: monthString = "October";
             break;
    case 11: monthString = "November";
             break;
    case 12: monthString = "December";
             break;
    default: monthString = "Invalid month";
             break;
}
System.out.println(monthString);
```

Introduction to Computer Science
Reichman University

Recitation 2

Control Flow - Loops

I will not argue with idiots on YouTube.
I will not argue with idiots on YouTube.
I will not argue with idiots on YouTube.
I will not argue with idiots on YouTube.
I will not argue with idiots on YouTube.

Loops - why and when to use each?

- While is more commonly used when we don't know the exact number of iterations. while is freer to use but there is more likelihood to have an infinite loop.
- For is more commonly used when we do know the exact number of iterations. for is has more restrictions to use but the structure help to avoid infinite loops.

Introduction to Computer Science
Reichman University

Recitation 2

Shorthand Operations

Shorthand Ops

- Shorthand operators in Java allow you to write more concise code for common operations and assignments.
- Advantages:
  - Improved code readability.
  - Concise expressions for common operations.
  - Can make your code more efficient and easier to maintain.

```
int y = 10;

y--; // Decrement by 1 (Equivalent to y = y - 1)

int b = 7;

b -= 6; // Subtracts 6 from 'b' (Equivalent
to b = b - 7)

int price = 20;

price /= 2; // Divides 'price' by 2 (Equivalent
to price = price / 2)
```

```
int x = 5;

x++; // Increment by 1 (Equivalent to x = x + 1)

int a = 5;

a += 3; // Adds 3 to 'a' (Equivalent to a = a + 3)

int total = 8;

total *= 5; // Multiplies 'total' by 5 (Equivalent to
total = total * 5)
```

Shorthand Ops - i++ Vs ++i ?

- There are some languages which uses ++i rather than i++.
- What is the difference between them?
- When you see ++i at line n, it means that there is an "imaginary" line between lines n-1 and n, which increases the value of i by 1.
  - int i = 4;
  - System.out.println(++i); // 5
  - System.out.println(i);   // 5

- When you see i++ at line n, it means that there is an "imaginary" line between lines n and n+1, which increases the value of i by 1.
  - int i = 4;
  - System.out.println(i++); // 4
  - System.out.println(i);   // 5
- Tip: always use i++ over ++i.

Question 5 - i++ Vs ++i

A

```
int n = 6;
for (int i = 0; i < n; i++) {
    System.out.println(i);
}
```

B

```
int n = 6;
for (int i = 0; i < n; ++i) {
    System.out.println(i);
}
```

C

```
int n = 6;
int i = 0;
while (i < n) {
    System.out.println(i++);
}
```

D

```
int n = 6;
int i = 0;
while (i < n) {
    System.out.println(++i);
}
```

- The following for segments of code are very similar.
- 3 of those will have the same output, one will have different output.
- Which one is it? Why?

Introduction to Computer Science
Reichman University

Recitation 2

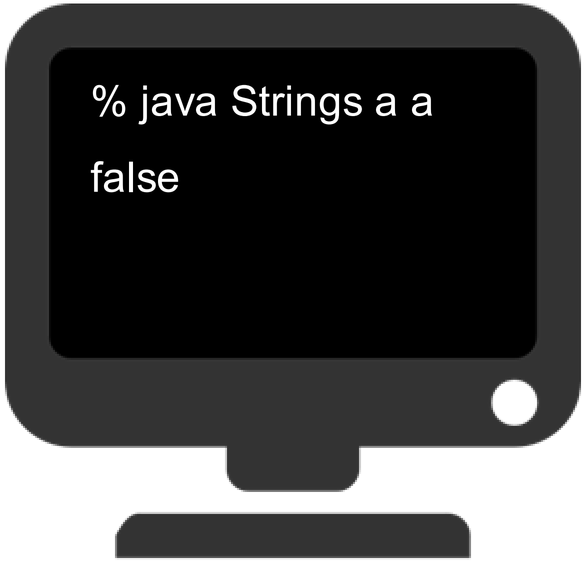
Control Flow - Loops

I will not argue with idiots on YouTube.
I will not argue with idiots on YouTube.
I will not argue with idiots on YouTube.
I will not argue with idiots on YouTube.
I will not argue with idiots on YouTube.

Question 5 – Password Validation

- When we sign up to a website it asks us to enter a password and then type it again to ensure the correct password is written.
- One problem, with **<u>strings not being primitive type</u>** it means that things don't always work like we expect them to.
  - String a = "a";
  - String b = "a";
  - System.out.println (a == b);
  - // true
- However, consider the following program to compare input strings:

```java
public class Strings {
        public static void main(String[] args){
                String a = args[0];
                String b = args[1];
                System.out.println(a == b);
        }
    }
```



```
% java Strings a a
false
```

Question 5: Strings - Revisited

- What happened there?
- When comparing strings using '==' java actually checks the where in the memory, both strings are stored, since the locations in the memory of the strings and not the same the answer is false. The value itself wasn't checked.
- Since the compiler could not know before hand that the command line arguments would be identical, they got different addresses.
- In general, it's bad practice to compare strings using '=='. Unless you're interested to compare their addresses, hence we will use '==' for primitive types only.
- When we do want to compare Strings we use string.equals(String other).
- Note: This "phenomenon" is not specifically with strings. Every non-primitive type in Java works that way.
- Let's write a program which checks if two strings are the same. Why? One of the key aspect of coding is making consistent code.

Question 5 - Solution

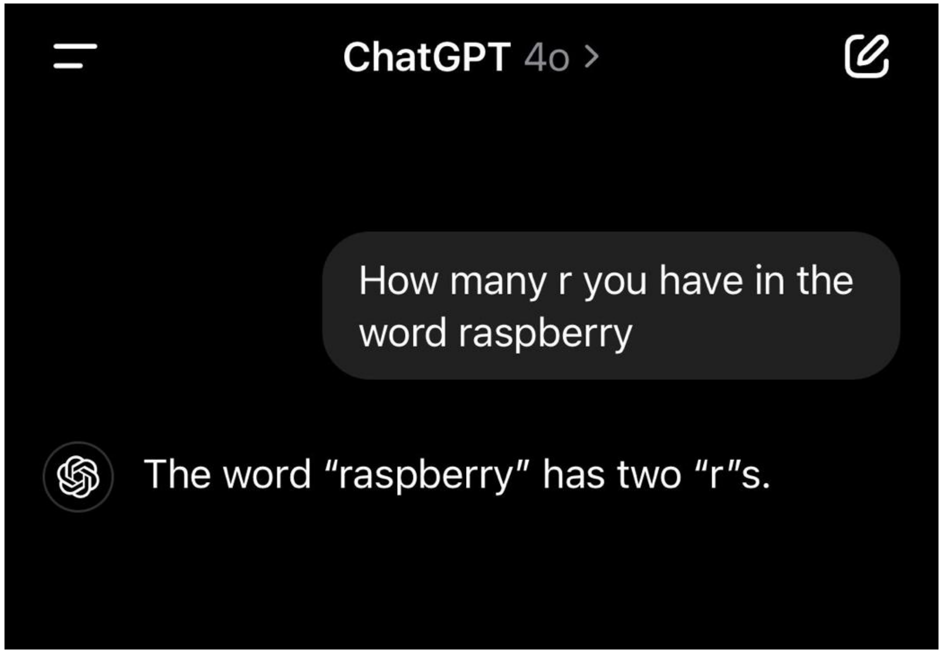
```
public class IsSamePassword {
        public static void main(String[] args){
                String a = args[0];
                String b = args[1];
                boolean ans = a.length() == b.length();
                for (int i = 0; i < a.length() && ans; i++){
                        ans = (a.charAt(i) == b.charAt(i));
                }
                System.out.println(ans);
        }
    }
}
```

Question 6 – Counting chars

- When we set up our passwords there are certain rules applied in order to make it “Strong Password”. One of the less used rules is not having the same char more than a certain number of times.
- In those time usually we ask for help from ChatGPT. Let's look what it answers on certain words.

Question 6 – Strings

- Who can tell me what is wrong here?
- The word raspberry has 3 "r"s
  - <u>r</u>aspbe<u>rr</u>y
- What happens here? The AI splits the word raspberry to 2 words ("rasp" and "berry"). And returns the more probable answer of the 2 options which in this case is 2.
- So we must build a function which will be able to work on all 3 words correctly.



How many r you have in the word raspberry

The word "raspberry" has two "r"s.

https://prompt.16x.engineer/blog/why-chatgpt-cant-count-rs-in-strawberry

The problem was discovered on strawberry, which was fixed. But the problem currently live in raspberry. So you can try it yourself.

Question 6 – Strings

- Design a program which does the following:
  - The program receives a string (will be called str for the question and a char (will be called ch in the question) from the users.
    - you may assume that the second command line argument is only one char.
    - Think how can you convert a String to char.
  - Then the program counts the number of appearances of ch in str
  - prints it
- Examples
  - % java CharRunCount rasp r
    - 1
  - % java CharRunCount berry r
    - 2
  - % java CharRunCount raspberry r
    - 3
  - % java CharRunCount raspberry R
    - 0
  - % java CharRunCount "raspberry juice" r
    - 3
  - % java CharRunCount Hello t
    - 0

Question 6 - Solution

```
public class CharRunCount {
        public static void main(String[] args){
            String word = args[0];
            char ch = args[1].charAt(0);
            int count = 0;
            for (int i = 0; i < word.length(); i++){
                if (word.charAt(i) == ch){
                      count++;
                 }
            }
            System.out.print("The char: " + ch + " appears " + count);
            System.out.println(" times in the word: " + word);
      }
}
```

Question 6, Expansion 1 – Counting chars

- When we set up our passwords there are certain rules applied to make it “Strong Password”. One of the less used rules is not having the same char more than a certain number of times **in Sequence**.
- Let's upgrade our program to handle **repeating chars in sequence only**, and check **for all chars**.

Question 6, Expansion 1 - Solution

```
public class CharRunCountInSequence {
        public static void main(String[] args){
            String word = args[0];
            int maxAmountOfRepeatingInSequence = Integer.parseInt(args[1]);
            int count = 0;
            int maxCount = 0;
            int index = 0;
            for (int i = 0; i < word.length(); i++){
                    char currentChar = word.charAt(i);
                    index = i;
                    count = 0;
                    while (word.charAt(index) == currentChar && index < word.length()){
                         index++;
                         count++;
                    }
                    maxCount = Math.max(count, maxCount);
            }
            if (maxCount <= maxAmountOfRepeatingInSequence) {
                    System.out.println("No char appeared more than " + maxAmountOfRepeatingInSequence + " times.");
            } else {
                    System.out.println("There is a char that exists more than" + maxAmountOfRepeatingInSequence
                                                 + " times in the word: " + word);
            }
        }
}
```

Question 7 – Strings

- We talked about the string method str.indexOf(char ch).
- Let's design a program which works similarly.
  - The program receives a 2 strings from the user.
  - Validate the second-string input (suppose to be a char) :
    - If it is an empty string : assign the value space (' ').
    - If it has one char or more : take the first char only.
  - The program finds the first index that the second validated string appears in the first string, if it does appear otherwise, shows the value -1.
- Note: In these kind of questions, when you are required to implement a function which exists. You may not use the original function or any other built-in functions which has a similar goal (unless specified otherwise), even if you learnt those functions in class.
- Therefore, we **cannot** use the function str.indexOf(char ch); or any other string methods which find a char in string.

Question 7 - Solution

```
public class FindCharInString {
        public static void main(String[] args){
          String source = args[0];
          char chr = args[1].length() > 0 ? args[1].charAt(0) : ’ ’;
          String temp = "Find the char: " + chr + "in the string: ";
          temp += source;
          System.out.println(temp);
          int result = -1; // if index not found return -1 as error control
          boolean flag = false;
          for (int index = 0; index < source.length() && !flag; index++){
                 if (chr == source.charAt(index)){
                          result = index;
                          flag = true;
                 }
        }
        System.out.println(result);
    }
}
```

Question 8 – Strings

- Sometimes you can combine previously solved questions in order to solve a new question (with slight changes)
- We will use two questions that we already solved today with minor changes to get the solution.

Note: A version of the next question appeared in Midterm 2022 (Q5 there for the curious)

- Design a program which receives a String and a char from the user
- then prints the number of every appearance of that char with the position it appears in the String
- Example:
  - % java AllIndexOf "hello world" l
    - (1,2), (2,3), (3,9)
  - % java AllIndexOf "hello worLd" l
    - (1,2), (2,3)
  - % java AllIndexOf "hello world" p
    - "" // empty string

Question 8 - Solution

```
public class AllIndexOf {
    public static void main(String[] args) {
           // Q6- charRunCount
           String word = args[0];
           char ch = args[1].charAt(0);
           int count = 0;
           for (int i = 0; i < word.length(); i++) {
                if (word.charAt(i) == ch){
                     count++;
                 }
           }
           System.out.println("The char: " + ch + " appears " + count + " time in the word: " +
                                                                               word);
           // Q7 - FindCharInString (with some adjustments)
           String temp = "Find all appearnces of the char: " + ch + "in the string: " + word;
           System.out.println(temp);
           int curCounter = 1;
           String res = "";
           for (int index = 0; index < word.length() && curCounter <= count; index++){
                if (ch == word.charAt(index)){
                     if (curCounter != 1){
                           res += ", "; // Aesthetics
                     }
                      // note curCounter value increased by 1 after the following is excecuted
                     res += "(" + (curCounter++) + "," + index + ")";
                }
           }
           System.out.println(res);
    }
}
```

Recitation 2

Variable Scope

I AM DAENERYS OF HOUSE TARGARYEN, THE FIRST OF HER NAME, QUEEN OF THE ANDALS AND THE FIRST MEN, THE UNBURNT, KHALEESI OF THE GREAT GRASS SEA, THE BREAKER OF CHAINS, MOTHER OF DRAGONS, ALSO KNOWN AS DAENERYS STORMBORN OF THE BLOOD OF OLD VALYRIA, MHYSA, THE SILVER QUEEN, BRIDE OF FIRE, THE DRAGON QUEEN, PRINCESS OF DRAGONSTONE, QUEEN OF MEEREEN, LADY REGNANT OF THE SEVEN KINGDOMS. THE QUEEN ACROSS THE SEA, THE BEGGAR QUEEN, CHILD OF THREE SISTER TO THE BEGGAR KING, SLAYER OF LIES, DAUGTHER OF DEATH AND PROTECTOR OF THE REALM.
OR YOU CAN SIMPLY CALL ME
DANY

Variable Scope

- In Java, each block is equal to a scope. Meaning, any segment of code contained within curly brackets is a scope.
- Scopes can be contained in other scopes (for example, nested ifs).
- A scope determines what variables are visible and accessible to other part of your program.
- The scope of a local variable is the rest of the block in which it was declared.
- Variables cannot be called or used outside their scope.
- The name of a local variable v cannot be used again in the same scope.

The “Khaleesi Rule” - Metaphor for scoping

- Assume that a given room is a scope, “Khaleesi” is the variable, “Khaleesi”’s reaction is the compiler (Not an official name but a way to remember)
  - If you try to address “Khaleesi” before she was properly introduced (declared) -> you disrespect her therefore “**dracarys**“ (compiler error).
  - If you try to introduce her properly twice in the same room -> you are wasting her time therefore “**dracarys**“.
  - If you have room inside a room and introduced her properly in the bigger room and afterwards try to introduce her properly again in smaller room -> you are wasting her time therefore “**dracarys**“.
  - If you have room inside a room and introduced her properly in the smaller room and try to address her in the bigger room -> you disrespect her therefore “**dracarys**“.
  - If you have room inside a room and introduced her properly in the smaller room and afterward try to introduce her properly in the bigger room -> OK.
  - If you have room inside a room and introduced her properly in the bigger room and afterwards try to address her in smaller room -> OK .

Example 1 & 2 - Scope

```
        public static void main (String[] args) {
                // a’s scope starts here
                int a = 10;
                while (true) {
                    int a = 100;
                    // Compilation error – a already ‘exists’ in this scope
                }
        }
              // a’s scope ends here, upon exiting ‘main’
```

```
        public static void main (String[] args) {
                while (true) {
                    // a’s scope starts here
                    int a = 100;
                }
                // a’s scope ends here
                System.out.println(a);
                // Compilation error – a doesn’t ‘exist’ in this scope
        }
```

Example 3 & 4 - Scope

```
public static void main (String[] args) {
        // a’s scope starts here
        int a = 10;
        while (true) {
            a = 100;
            // that’s fine, just a reassignment
        }
}
      // a’s scope ends upon exiting ‘main’
```

```
public static void main (String[] args) {
            while (true) {
                       // a’s scope starts here
                       int a = 100;
                       // note that each iteration uses a new declaration.
            }
           // a’s scope ends upon exiting the loop
            int a = 100;
}
```

Recitation 2

Extra