Review Topics

Computer Hardware

Input

1. What class is commonly used to get input from the user in Java?
2. How does the nextLine() method work in Java's Scanner class?
3. What is the purpose of the nextInt() method in Java's Scanner class?
4. How can you read a string from the user in Java?
5. What happens if the user enters an invalid value when using Scanner?
6. How do you read a floating-point number using Scanner in Java?
7. What does the System.in stream do in Java?
8. How do you ask the user for input using Scanner in Java?
9. How do you handle multiple inputs on the same line using Scanner?
10. What is the difference between next() and nextLine() in Scanner?

Output

1. What is the difference between System.out.print() and System.out.println() in Java?
2. How do you print a string to the console without moving to a new line in Java?
3. What is the result of using System.out.println() in Java?
4. How do you combine multiple values in one System.out.println() statement?
5. How do you print an integer and a string on the same line in Java?
6. How can you print formatted text in Java using printf?
7. How can you use escape sequences in print statements (like \n for newline)?
8. What would happen if you print a null value in Java?
9. How do you print a variable value in System.out.println()?
10. How do you use concatenation in System.out.println()?

Memory

1. What does RAM stand for, and what is its role in a computer?
2. What is the difference between volatile and non-volatile memory?
3. What does ROM stand for, and why is it used in computers?
4. What is the difference between primary and secondary memory?
5. How does the CPU use registers for processing?
6. What is the role of the Control Unit in the CPU?
7. Why is RAM faster than secondary storage like hard drives?
8. What happens to the data in RAM when the computer is turned off?
9. How is data stored temporarily in a computer?
10. What is the purpose of the ALU in a computer system?

Data Types

1. What is a byte in Java, and what kind of data does it store?
2. How much memory does an int occupy in Java?
3. What is the difference between an int and a long in Java?
4. What is the size of a char in Java?
5. How do you store a decimal number in Java?
6. What is the maximum range of a short in Java?
7. What is the default value of a boolean in Java?
8. How does a double differ from a float in Java?
9. What is the maximum value of an int in Java?
10. How do you convert a string to an integer in Java?

Operators

1. What does the + operator do in Java?
2. What is the purpose of the % operator in Java?
3. How does the ++ operator work in Java?
4. What is the difference between ++i and i++ in Java?
5. How do you subtract two numbers in Java?
6. How do you divide two integers in Java?
7. What happens if you divide by zero in Java?
8. How can you use the modulus operator (%) in Java to check if a number is even or odd?
9. What is the result of 7 % 3 in Java?
10. How do you increment a variable by 1 in Java?

Control Structures

1. How do you write an if statement in Java?
2. What is the difference between an if statement and an if-else statement in Java?
3. How do you use an if-else if statement in Java?
4. What happens if the condition in an if statement is false?
5. How do you use a switch statement in Java?
6. How does the break statement work inside a switch case in Java?
7. What is the purpose of the else clause in an if-else statement?
8. How do you check multiple conditions in an if statement in Java?
9. How do you write a nested if statement in Java?
10. What is the difference between && (AND) and || (OR) operators in an if statement?

Number Systems

1. What base system is used in the decimal number system?
2. What digits are used in binary?
3. What is the base of the octal number system?

4. How do you convert a decimal number to binary?
5. What is the hexadecimal system used for in computing?
6. What are the digits used in the hexadecimal system?
7. How do you convert a binary number to decimal?
8. What is the base of the hexadecimal number system?
9. How do you convert a decimal number to hexadecimal?
10. How do you convert a binary number to octal?

Logical Operators

1. What does the && operator do in Java?
2. What is the result of true && false in Java?
3. How do you use the || operator in Java?
4. What is the result of true || false in Java?
5. How does the ! operator work in Java?
6. What does the expression !false evaluate to in Java?
7. How would you write a conditional statement using && in Java?
8. What is the difference between && and || in Java?
9. How do you negate a boolean value in Java?
10. How does a logical expression with multiple conditions evaluate in Java?

String Methods

1. How do you find the length of a string in Java?
2. What does the charAt() method do in Java?
3. How can you find the index of a character in a string in Java?
4. How do you get a substring from a string in Java?
5. How do you convert a string to uppercase in Java?
6. How do you convert a string to lowercase in Java?
7. What does the compareTo() method do in Java?
8. How do you check if two strings are equal in Java?
9. How can you check if a string contains a specific character in Java?
10. How do you replace a character in a string in Java?

Arduino Functions

1. What is the purpose of the setup() function in Arduino?
2. What is the purpose of the loop() function in Arduino?
3. How do you use the digitalWrite() function in Arduino?
4. What is the difference between digitalWrite() and analogWrite() in Arduino?
5. How do you turn an LED on using digitalWrite() in Arduino?
6. How do you control the brightness of an LED using analogWrite() in Arduino?
7. What does the pinMode() function do in Arduino?
8. How do you use delay() in Arduino?

9. How can you use digitalRead() in Arduino to read a button press?
10. What is the maximum value you can pass to analogWrite() in Arduino?

LED and Resistor Color Codes

1. What is the color code for a 10k ohm resistor?
2. What does the color red represent on a resistor's color code?
3. How many ohms does a resistor with the color code "Green, Black, Red" have?
4. What is the multiplier for the color code orange on a resistor?
5. How do you determine the resistance value of a resistor using its color code?
6. What is the color for a 5% tolerance resistor?
7. How many digits does the color code on a resistor contain?
8. How do you identify the anode and cathode of an LED?
9. What is the value of a resistor with the color code "Blue, Black, Brown"?
10. What does the color brown indicate for a resistor’s tolerance?

Math.random()

1. How do you generate a random number between 0 and 1 using Math.random()?
2. How do you generate a random integer between 0 and 99 using Math.random()?
3. How can you use Math.random() to generate a random number between 1 and 100?
4. What is the range of values returned by Math.random() in Java?
5. How do you generate a random number within a specific range using Math.random()?
6. How do you use Math.random() to simulate a dice roll (1-6)?
7. How can you round a random float generated by Math.random()?
8. How can you generate a random number with a specific distribution using Math.random()?
9. How do you convert a random double from Math.random() to an integer?
10. What type of value does Math.random() return in Java?

Answer Key

Computer Hardware

Input

1. `Scanner` class is commonly used to get input from the user in Java.
2. The `nextLine()` method reads an entire line of input, including spaces.
3. `nextInt()` reads an integer input from the user.
4. Use `scanner.nextLine()` to read a string from the user in Java.
5. If the user enters an invalid value, an `InputMismatchException` is thrown.
6. Use `scanner.nextFloat()` to read a floating-point number.
7. `System.in` is an InputStream that represents standard input, typically the keyboard.
8. Use `scanner.next()` or `scanner.nextLine()` to prompt the user for input.
9. Use `scanner.nextLine()` to read the whole line, or `scanner.next()` for space-separated tokens.
10. `next()` reads the next token (space-separated), while `nextLine()` reads the entire line.

Output

1. `System.out.print()` prints without a newline, while `System.out.println()` prints with a newline at the end.
2. To print without a newline, use `System.out.print()`.
3. `System.out.println()` prints a value followed by a newline character.
4. Concatenate values with `+` in `System.out.println()`.
5. Use `+` to combine variables with strings in `System.out.println()`.
6. Use `System.out.printf()` to print formatted text.
7. Escape sequences like `\n` (newline) can be used inside strings to control formatting.
8. Printing `null` will result in the string `"null"`.
9. Use `System.out.println(variableName)` to print a variable's value.
10. Concatenate strings and variables with `+` in `System.out.println()`.

Memory

1. RAM stands for Random Access Memory; it is temporary storage used for fast data access.
2. Volatile memory loses its data when power is turned off, while non-volatile retains it.
3. ROM stands for Read-Only Memory; it stores data permanently, like firmware.

4. Primary memory includes RAM and cache; secondary memory includes hard drives and SSDs.
5. Registers hold data temporarily for fast access during processing.
6. The Control Unit directs operations in the CPU by interpreting instructions.
7. RAM is faster because it stores frequently accessed data for quick retrieval.
8. Data in RAM is lost when power is turned off.
9. Temporary data is stored in RAM for processing.
10. The ALU (Arithmetic Logic Unit) performs mathematical and logical operations.

Data Types

1. A byte is a data type that holds an 8-bit value.
2. An `int` occupies 4 bytes (32 bits).
3. An `int` has a smaller range than a `long` (which occupies 8 bytes).
4. A `char` in Java is 2 bytes (16 bits).
5. A decimal number is stored in a `float` or `double`.
6. A `short` can hold values between -32,768 and 32,767.
7. The default value of a boolean is `false`.
8. A `double` is more precise and has a wider range than a `float`.
9. The maximum value of an `int` is 2^31 - 1 (2,147,483,647).
10. Use `Integer.parseInt(string)` to convert a string to an integer.

Operators

1. The + operator adds two numbers, or concatenates strings.
2. The % operator returns the remainder of division.
3. The ++ operator increments a variable by 1.
4. `++i` increments first, then returns the value, while `i++` increments after returning the value.
5. Use `-` to subtract two numbers in Java.
6. Use `/` to divide two integers in Java.
7. Dividing by zero throws an `ArithmeticException` in Java.
8. Use `% 2 == 0` to check if a number is even, or `% 2 == 1` to check if it's odd.
9. 7 % 3 equals 1.
10. Use `i++` or `++i` to increment a variable by 1.

Control Structures

1. `if (condition) { // code }` is the syntax for an if statement.
2. An `if` statement executes if the condition is true; `if-else` executes one block of code if true, and another if false.

3. Use `if (condition1) { // code1 } else if (condition2) { // code2 }` for multiple conditions.
4. If the condition is false, the code inside the if block will not execute.
5. A switch statement allows testing of multiple conditions based on a variable's value.
6. The `break` statement exits a switch case.
7. The else clause executes if the `if` condition is false.
8. Use logical operators like `&&` or `||` to check multiple conditions.
9. A nested if is an if statement inside another if statement.
10. `&&` means both conditions must be true, while `||` means at least one condition must be true.

Number Systems

1. The decimal system is base 10.
2. Binary uses only the digits 0 and 1.
3. Octal uses base 8, with digits 0-7.
4. To convert decimal to binary, repeatedly divide by 2, recording the remainders.
5. Hexadecimal (base 16) is used in computing for concise representation of binary data.
6. Hexadecimal uses digits 0-9 and letters A-F.
7. To convert binary to decimal, multiply each bit by 2 raised to the power of its position and sum them.
8. Hexadecimal has a base of 16.
9. To convert decimal to hexadecimal, divide by 16 and record the remainders.
10. To convert binary to octal, group the binary digits in sets of three, then convert each group to an octal digit.

Logical Operators

1. The && operator checks if both conditions are true.
2. `true && false` evaluates to `false`.
3. The || operator checks if at least one condition is true.
4. `true || false` evaluates to `true`.
5. The ! operator negates a boolean value (turns `true` to `false`, or `false` to `true`).
6. `!false` evaluates to `true`.
7. Use `if (condition1 && condition2)` to check multiple conditions.
8. `&&` returns `true` only if both conditions are true, while `||` returns `true` if at least one condition is true.
9. Use `!` to negate a boolean value.
10. A logical expression with multiple conditions evaluates based on the rules of AND (`&&`) and OR (`||`).

String Methods

1. Use `string.length()` to find the length of a string.
2. `charAt(i)` returns the character at the given index `i`.
3. `indexOf(c)` returns the index of the first occurrence of `c` in the string.
4. `substring(start, end)` returns the substring from index `start` to `end - 1`.
5. `toUpperCase()` converts all characters to uppercase.
6. `toLowerCase()` converts all characters to lowercase.
7. `compareTo()` compares two strings lexicographically.
8. Use `string.equals()` to check if two strings are equal.
9. Use `string.contains(c)` to check if a string contains a specific character.
10. Use `string.replace(oldChar, newChar)` to replace a character in a string.

Arduino Functions

1. The `setup()` function is called once when the program starts and is used to initialize settings.
2. The `loop()` function runs repeatedly and is used for continuous actions.
3. `digitalWrite(pin, value)` sets the digital output of a pin to `value` (HIGH or LOW).
4. `digitalWrite()` is used for binary output, while `analogWrite()` is for PWM (pulse-width modulation) output.
5. Use `digitalWrite(pin, HIGH)` to turn on an LED.
6. Use `analogWrite(pin, value)` with a value between 0-255 to control LED brightness.
7. `pinMode(pin, mode)` sets the mode of a pin (INPUT or OUTPUT).
8. `delay(milliseconds)` pauses the program for the specified number of milliseconds.
9. `digitalRead(pin)` reads the digital input from a pin.
10. `analogWrite()` can take values between 0 and 255 to control the voltage output.

LED and Resistor Color Codes

1. A 10k ohm resistor has the color code "Brown, Black, Orange".
2. Red represents 2 in the resistor color code.
3. "Green, Black, Red" equals 5,000 ohms or 5k ohms.
4. Orange represents a multiplier of 10^3.
5. Use the color codes to identify the resistor's value, where the first two colors are digits and the third is the multiplier.
6. The color gold or silver represents 5% tolerance.
7. A resistor color code consists of four or five bands.
8. The longer leg is the anode (positive), and the shorter leg is the cathode (negative).

9. "Blue, Black, Brown" equals 1,000 ohms or 1k ohms.
10. The tolerance band is usually gold (5%) or silver (10%).

Math.random()

1. `Math.random()` generates a random number between 0 (inclusive) and 1 (exclusive).
2. To generate a random integer between 0 and 99, use `Math.floor(Math.random() * 100)`.
3. To generate a number between 1 and 100, use `Math.floor(Math.random() * 100) + 1`.
4. `Math.random()` returns a floating-point number between 0 (inclusive) and 1 (exclusive).
5. To generate a number within a specific range, use `Math.random() * (max - min) + min`.
6. `Math.floor(Math.random() * 6) + 1` simulates a dice roll.
7. To round a random float, use `Math.round()`.
8. Use the `Math.random()` output as a basis for custom distributions.
9. To convert a random double to an integer, use `Math.floor()`.
10. `Math.random()` returns a `double`.