CS1 Midterm Study Guide - CVHS v1 2024, By: Vir Sanghavi

---

1. Computer Hardware

Input

- **Definition:** Process of sending data or instructions to a computer for processing. Input can come from hardware devices (e.g., keyboard, mouse) or software (e.g., API calls).
- **Types of Input Devices:**
  - **Textual Input:** Keyboard for typing data into programs.
  - **Graphical Input:** Mouse for clicking, dragging, or selecting.
  - **Sensory Input:** Sensors in embedded systems or IoT devices (e.g., temperature sensors).

Programming Example (Java):

```
import java.util.Scanner;



public class InputExample {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter your name: ");

        String name = scanner.nextLine();

        System.out.println("Hello, " + name + "!");

    }

}
```

- **Explanation:** Scanner reads input from the user. `nextLine()` captures a line of input as a string.

Output

- **Definition:** Process of sending processed data from the computer to the outside world (e.g., monitor, speaker, printer).
- **Types of Output Devices:**
  - **Monitor:** For text, images, and video.

    - **Speaker:** For audio signals.
    - **Actuators:** For mechanical outputs in robotics.

Programming Example:

```
public class OutputExample {

    public static void main(String[] args) {

        System.out.println("This is an output example.");

    }

}
```

- **Explanation:** `System.out.print()` outputs text without a new line. `System.out.println()` outputs text with a new line.

Memory

- **Definition:** Where the computer stores data.
- **Types:**
    - **Primary Memory:**
        - **Volatile:** Temporary storage (e.g., RAM).
        - **Non-volatile:** Permanent storage (e.g., ROM).
    - **Secondary Memory:** Hard drives (HDD), solid-state drives (SSD).
- **Key Memory Terms:**
    - **Bit:** Smallest memory unit; a 0 or 1.
    - **Byte:** 8 bits, used to store characters or small numbers.
    - **RAM:** Temporary memory for active processes.
    - **ROM:** Permanent, read-only memory for firmware.

RAM (Random Access Memory)

- **Definition:** Volatile memory used for running processes.
- **Characteristics:**
    - Faster than storage memory (HDD/SSD).
    - Data is erased when the computer is turned off.

EPROM (Erasable Programmable Read-Only Memory)

- **Definition:** Non-volatile memory that can be erased and reprogrammed.
- **Use Cases:** Firmware storage in embedded systems.

Microprocessor

- **Definition:** Central unit responsible for executing instructions (e.g., in smartphones, embedded devices).

CPU (Central Processing Unit)

- **Definition:** The "brain" of the computer, where most calculations are performed.
- **Components:**
  - **Control Unit (CU):** Directs data flow.
  - **Arithmetic Logic Unit (ALU):** Handles calculations and logic.
  - **Registers:** Small, fast memory inside the CPU.

ALU (Arithmetic Logic Unit)

- **Definition:** Subsystem of CPU that performs arithmetic and logical operations.

Example in Java:
int result = 5 + 3;

---

2. Data Types

Primitive Data Types
Java has 8 primitive data types.
Memorization Strategy: BSILCFDB (Byte, Short, Int, Long, Char, Float, Double, Boolean).

Data Type	Size	Range	Example
byte	8 bits	-128 to 127	`byte b = 100;`
short	16 bits	-32,768 to 32,767	`short s = 30000;`
int	32 bits	-2^31 to 2^31-1	`int x = 1000;`
long	64 bits	-2^63 to 2^63-1	`long l = 100000L;`

float	32 bits	Up to 7 decimal digits	`float pi = 3.14f;`
double	64 bits	Up to 15 decimal digits	`double gpa = 4.0;`
char	16 bits	Unicode (0 to 65,535)	`char letter = 'A';`
boolean	1 bit	true or false	`boolean isActive = true;`

String

- **Definition:** A sequence of characters (objects in Java, not primitive types).

Example:
String greeting = "Hello, World!";

-

---

3. Operators

Arithmetic Operators

Operator	Description	Example	Result
+	Addition	5 + 3	8
-	Subtraction	5 - 3	2
*	Multiplication	5 * 3	15

/	Division	5 / 2	2
%	Modulus	5 % 2	1

Increment/Decrement Operators

Operator	Description	Example	Result
++	Increment by 1	`x++`	Adds 1 to x
--	Decrement by 1	`x--`	Subtracts 1 from x

---

4. Control Structures

If-Else Statements

Syntax:

```
 if (condition) {

    // Code block if condition is true

} else if (anotherCondition) {

    // Code block if another condition is true

} else {

    // Code block if none of the conditions are true

}
```

Example:

```
 int age = 18;

if (age < 18) {

    System.out.println("Minor");

} else if (age == 18) {

    System.out.println("Exactly 18");

} else {

    System.out.println("Adult");

}
```

---

5. Number Systems

Introduction to Number Systems

- A number system is a standardized way to represent numbers.

Number System	Base	Digits/Symbols
Decimal	10	0, 1, 2, 3, 4, 5, 6, 7, 8, 9
Binary	2	0, 1
Octal	8	0, 1, 2, 3, 4, 5, 6, 7
Hexadecimal	16	0-9, A, B, C, D, E, F (A=10...)

Conversion Between Number Systems

- **Decimal ↔ Binary**
    - Decimal to Binary: Divide by 2, write remainder, reverse.
    - Binary to Decimal: Multiply each bit by 2^n, sum results.
- **Decimal ↔ Octal**
    - Decimal to Octal: Divide by 8, write remainder, reverse.
    - Octal to Decimal: Multiply each digit by 8^n, sum results.
- **Decimal ↔ Hexadecimal**
    - Decimal to Hexadecimal: Divide by 16, write remainder (0-9 or A-F), reverse.
    - Hexadecimal to Decimal: Multiply each digit by 16^n, sum results.
- **Binary ↔ Hexadecimal**
    - Binary to Hexadecimal: Group binary digits into sets of 4, convert to hexadecimal.
    - Hexadecimal to Binary: Convert each hexadecimal digit to 4-bit binary.

---

6. Logical Operators

Logical operators are used to evaluate boolean expressions.

Operator	Description	Example	Result
&&	Logical AND	true && false	false
||	Logical OR	true || false	true
!	Logical NOT	!true	false

Example:

boolean isRaining = true;

boolean hasUmbrella = false;

```
if (isRaining && hasUmbrella) {
    System.out.println("You're dry!");
} else if (isRaining || hasUmbrella) {
    System.out.println("You might stay dry.");
} else {
    System.out.println("You're getting wet.");
}
```

---

7. String Methods

Method	Description	Example	Result
`length()`	Returns the length of the string	`"hello".length()`	5
`charAt(i)`	Returns the character at index i	`"hello".charAt(1)`	'e'
`indexOf(c)`	Returns the index of the first occurrence of c	`"hello".indexOf('l')`	2
`substring(start)`	Returns substring from start to the end	`"hello".substring(2)`	"llo"
`substring(start, end)`	Returns substring from start to end-1	`"hello".substring(1, 4)`	"ell"

`toUpperCase()`	Converts to uppercase	`"hello".toUpperCase()`	"HELLO"
`toLowerCase()`	Converts to lowercase	`"HELLO".toLowerCase()`	"hello"
`compareTo(s)`	Compares strings lexicographically	`"apple".compareTo("banana")`	Negative value

---

8. Arduino Functions

Core Functions

`setup()`: Runs once at the start. Initializes pins or variables.

```
void setup() {
  pinMode(13, OUTPUT); // Set pin 13 as OUTPUT
}
```

`loop()`: Runs continuously after `setup()`.

```
void loop() {
  digitalWrite(13, HIGH); // Turn LED on
  delay(1000);            // Wait 1 second
  digitalWrite(13, LOW);  // Turn LED off
  delay(1000);            // Wait 1 second
}
```

- **`digitalWrite()` and `analogWrite()`**

`digitalWrite(pin, value)`: Set a digital pin to HIGH or LOW.

```
digitalWrite(13, HIGH); // Turn pin 13 on
digitalWrite(13, LOW);  // Turn pin 13 off
```

`analogWrite(pin, value)`: Output PWM signal (0-255) to a pin.
analogWrite(9, 128); // 50% duty cycle

---

9. LED and Resistor Color Codes

LED Basics

- **Pins:**
  - Long pin = Anode (+)
  - Short pin = Cathode (-)

Resistor Color Code Chart

Color	Digit(1&2)	Multiplier	Tolerance
Black	0	10^0	-
Brown	1	10^1	±1%
Red	2	10^2	±2%
Orange	3	10^3	-
Yellow	4	10^4	-

Green	5	10^5	±0.5%
Blue	6	10^6	±0.25%
Violet	7	10^7	±0.1%
Gray	8	10^8	±0.05%
White	9	10^9	-
Gold	-	**10^ -1**	**±5%**
Silver	-	**10^ -2**	**±10%**

---

10. Math.random()

Generates a random double between 0.0 (inclusive) and 1.0 (exclusive).
You can scale it for integers:

int randomInt = (int) (Math.random() * 100); // Random number 0-99