# OODP Workshop 8

1. What is the difference between Array and ArrayList?

In Java, Arrays and ArrayLists are both used to store collections of elements, but they have some fundamental differences. Here's a table that highlights the key differences between them with small examples:

|  |  |  |
| --- | --- | --- |
| **Aspect** | **Array** | **ArrayList** |
| Definition | Arrays are static data structures with a fixed size. | ArrayLists are dynamic data structures that can grow or shrink in size. |
| Example | int[] arr = new int[5]; | ArrayList<Integer> list = new ArrayList<>(); |
| Size | The size of an array is fixed and determined during its creation. | The size of an ArrayList can change dynamically at runtime. |
| Example | int[] numbers = {1, 2, 3, 4, 5}; | list.add(1); list.add(2); list.add(3); |
| Memory | Arrays are more memory-efficient as they store elements contiguously in memory. | ArrayLists have some overhead due to dynamic resizing and extra object creation. |
| Adding/Removing Elements | Arrays do not provide built-in methods to add or remove elements. You need to create a new array or use System utility methods. | ArrayLists provide built-in methods like add(), remove(), addAll(), and removeAll() to manipulate elements. |
| Example | (Adding an element is not straightforward) | list.add(4); list.remove(2); |
| Access Time | Arrays provide constant-time access to elements using indexes. | ArrayLists also provide constant-time access to elements using indexes, but with some overhead due to additional operations. |
| Example | int value = numbers[2]; // Constant-time access | int value = list.get(2); // Constant-time access |
| Primitives vs. Objects | Arrays can store both primitive data types (e.g., int, double) and objects. | ArrayLists can only store objects (including wrapped primitive types like Integer, Double). |
| Example | int[] arr = {1, 2, 3}; | ArrayList<Integer> list = new ArrayList<>(Arrays.asList(1, 2, 3)); |

1. Create an arrayList named List which will store string values and demonstrate add names, delete names, insert names of your friends into it and print them using loop.

List<String> friendsList = new ArrayList<>();

// Add names to the ArrayList

friendsList.add("John");

friendsList.add("Jane");

friendsList.add("Bob");

friendsList.add("Alice");

// Remove a name from the ArrayList

friendsList.remove("Bob");

// Insert a name at a specific index

friendsList.add(2, "David");

// Print the names using a loop

for (String name : friendsList) {

System.out.println(name);

}

1. Think about yourself as an object, identify attributes and behavior that you can possess in one payroll system.

In an object-oriented payroll system, If I think myself as an employee can be represented as an object with various attributes and behaviors. Some potential attributes and behaviors that an employee object could possess:

Attributes (Member Variables):

1. name (String): The name of the employee.
2. employeeId (String or int): A unique identifier for the employee.
3. dateOfBirth (Date or LocalDate): The date of birth of the employee.
4. address (String or an Address object): The address of the employee.
5. phoneNumber (String): The phone number of the employee.
6. email (String): The email address of the employee.
7. department (String or a Department object): The department the employee belongs to.
8. designation (String): The job title or designation of the employee.
9. dateOfJoining (Date or LocalDate): The date when the employee joined the company.
10. salary (double or Big Decimal): The base salary of the employee.
11. paymentMethod (String or an enum): The payment method (e.g., bank transfer, check, cash) for the employee's salary.
12. bankDetails (BankDetails object): The bank account details for salary transfer.
13. leaveBalance (int): The number of remaining leave days for the employee.
14. employeeType (String or an enum): The type of employee (e.g., full-time, part-time, contract).
15. taxRate (double): The tax rate applicable to the employee's income.

Behaviors (Methods):

1. calculateSalary(): Calculates the salary for the employee based on their base salary, allowances, deductions, and other factors.
2. applyLeave(int daysToApply): Updates the employee's leave balance by deducting the requested number of days.
3. updatePersonalInfo(...): Updates the personal information (e.g., address, phone number, email) of the employee.
4. changeDepartment(Department newDepartment): Changes the department of the employee.
5. promoteEmployee(String newDesignation, double newSalary): Promotes the employee to a new designation with a new salary.
6. terminateEmployee(Date terminationDate): Terminates the employee's employment and updates their status accordingly.
7. generatePayslip(): Generates a payslip for the employee with details like salary, deductions, and net pay.
8. calculateTaxDeduction(): Calculates the tax deduction based on the employee's salary and applicable tax rates.
9. updateBankDetails(BankDetails newBankDetails): Updates the employee's bank account details for salary transfer.
10. requestLeave(int daysToRequest): Requests a specific number of leave days for the employee.
11. Think about a product that you have in front of you, find out attributes and behavior that can possessed by this product.

Let's think about a laptop computer that is in front of me as an object and identify its potential attributes and behaviors.

Attributes (Member Variables):

1. brand (String): The brand name of the laptop (e.g., Dell, HP, Lenovo).
2. model (String): The specific model name or number of the laptop.
3. processor (String or Processor object): The processor or CPU model of the laptop.
4. ram (int): The amount of RAM (Random Access Memory) installed in the laptop, usually measured in gigabytes (GB).
5. storage (int or Storage object): The storage capacity of the laptop's hard drive or solid-state drive (SSD), usually measured in gigabytes (GB) or terabytes (TB).
6. screenSize (double or float): The diagonal size of the laptop's display screen, typically measured in inches.
7. resolution (String or Resolution object): The resolution of the laptop's display screen (e.g., 1920 x 1080, 2560 x 1600).
8. operatingSystem (String or OperatingSystem object): The operating system installed on the laptop (e.g., Windows, macOS, Linux).
9. battery (Battery object): The battery capacity and remaining charge level of the laptop.
10. weight (double or float): The weight of the laptop, usually measured in kilograms (kg) or pounds (lb).
11. ports (List or Array of Port objects): The available ports on the laptop (e.g., USB, HDMI, Thunderbolt).
12. webcam (boolean or Webcam object): Indicates whether the laptop has an integrated webcam or not.
13. touchscreen (boolean): Indicates whether the laptop has a touchscreen display or not.
14. price (double or float): The price or cost of the laptop.

Behaviors (Methods):

1. powerOn(): Turns on the laptop and boots up the operating system.
2. powerOff(): Shuts down the laptop safely.
3. sleep() or hibernate(): Puts the laptop into a low-power state (sleep or hibernate mode).
4. installSoftware(Software software): Installs a specific software application on the laptop.
5. uninstallSoftware(Software software): Uninstalls a software application from the laptop.
6. connectToWifi(String ssid, String password): Connects the laptop to a Wi-Fi network using the provided SSID and password.
7. disconnectFromWifi(): Disconnects the laptop from the currently connected Wi-Fi network.
8. openApplication(String appName): Opens or launches a specific application installed on the laptop.
9. closeApplication(String appName): Closes or terminates a running application on the laptop.
10. adjustBrightness(int level): Adjusts the brightness level of the laptop's display screen.
11. adjustVolume(int level): Adjusts the volume level of the laptop's speakers or audio output.
12. updateSoftware(): Checks for and installs available software updates or operating system updates.
13. checkBatteryLevel(): Retrieves and displays the current battery level of the laptop.
14. chargeBattery(): Charges the laptop's battery when connected to a power source.