

Manipulating and creating arrays of objects is a crucial skill for any software developer, especially when dealing with complex datasets. In many real-world applications, you'll need to manage collections of related objects efficiently.

This lab will enable you to put the knowledge you have gained into practice by guiding you through creating and manipulating arrays of objects. It reinforces learning from previous courses and will use different classes along with iterative constructs to create objects, populate arrays, and perform operations on them. Are you ready? Let's dive in.

Remember

An 'array of objects' refers to a data collection in an array where each element is an object of a specific class. This means that these arrays can store multiple instances of a class.

Imagine you are developing a student record management system for a university. Organizing, accessing, and manipulating student data is essential for maintaining accurate and up-to-date records, such as exam results. The university is launching a new platform to manage student records efficiently.

You have the following data:

- *Students* [] size is 5 in the *Main.java* class
- Features: *rollNumber*, *name*, *age* in *Student* class

For this lab, you can assume that you have already set up a project and classes in Java.

Goal

Can you build the student record management system's core functionality, starting with organizing student records from a sample of student data? You will create objects from different classes, populate an array by storing objects in arrays, and perform operations on them.




Note: When you encounter this icon, it's time to get into your IDE and start coding!

Under the *com.lesson.lab* package is a class named *Main.java*. You will use this class as the *Main.java* class to write the primary methods and update the code-related array, which is the essential operation using the *Student* class.

Open the starter code for from the *src* folder, in which you'll find the following set up:

Student class which includes the attributes, constructor, and getter/setter methods. Additionally, you will find the *Main* class with predefined method structure.

 It's time to get coding!

Use a predefined array named *students* that can hold 5 *student* objects.

- TODO 1: create the other 4 objects of the student class and assign them to the array of *Students*.
- TODO 2: uncomment the method call once you have completed the mentioned tasks.

```
Class Main{

    public static void main(String args[]){

        // TODO 1: create other 4 objects of the student class and assign them to the
        array Students

        // TODO 2: uncomment the method call once you have completed the mentioned
        tasks

    }


}
```

Remember

In the constructor, you should pass values in the same order defined in the class.

Imagine a situation where a student has graduated or transferred to another university, and you need to remove their record from the student management system. Efficiently managing the student records, including the ability to remove outdated or irrelevant entries, is crucial for maintaining the integrity of the database.

In this task, you will write the logic to remove a student from the array of student records. This involves creating a new array with a size one less than the original array and copying all the elements except the one that needs to be removed.

 It's time to get coding!

Use the predefined method *removeElement* method in the *Main.java* class.

- TODO 3: create a new array with a length less than the existing one. For example, if your existing array is *students*, the new array's size will be *students.length - 1*.
- TODO 4 : use a for loop to copy all elements from the existing array to the new array, except for the element at the index you want to remove.
- TODO 5: if needed, add the new array back to your original array variable.
- TODO 6: call the *printArray* method and pass "Remove" and new array.

```
Class Main{

    public static void removeElement(int indexToDelete, Student[] originalArray){

        // TODO 3: create a new array with a length less than the existing one

        //TODO 4: use a for loop to copy all elements from the existing array to
the new array

        //TODOs continue

    }

}
```


Reflection point

This is one possible solution to remove an element. What other ways are there?

Hint: Unlike modification, deletion affects the array size because arrays are not dynamically resizable. Instead of '-1', what could you do?

Suppose a new student has enrolled at the university, and you need to add their record to the student management system. This task involves expanding the current array to accommodate the new student record and ensuring the data is correctly inserted into the array.

In this task, you will write the logic to add a new student to the array of student records. This involves creating a new array with a size one greater than the original array and copying all the elements to the new array before adding the new student object.

 It's time to get coding!

Use the predefined method `addElement` method in the *Main.java* class.

- TODO 7: define a new array with a length of `students.length + 1`. This ensures there is space for the new element.
- TODO 8: use a for loop to copy all elements from the existing students array to the new array. Iterate over each element and assign it to the corresponding index in the new array.
- TODO 9: create a *newStudent* object and assign it to the last index of the new array, which is `newArray[newArray.length - 1]`.
- TODO 10: call the `printArray` method and pass "Add" and new array.

```
Class Main{

    public static void addElement(Student newStudent, Student[] originalArray){

        //TODO 7: define a new array with a length of students.length + 1

        // TODO 8: use a 'for loop' to copy all elements from the existing students
        array to the new array


    }

}
```

As you work through these tasks, consider the efficiency and potential overhead of adding elements to an array. Since arrays have a fixed size, each addition requires creating a new array and copying all existing elements, which can be resource-intensive. Reflect on how this impacts performance and explore alternative data structures that might offer more efficient solutions for dynamic data management.

You have already learned how to remove and add students to the array. Now, it's important to understand how to update existing records. For this, you can use the `updateElement` method in the *Main.java* class.

Imagine a student has legally changed their name, or their age has been incorrectly recorded in the system, and you need to update their record. In this task, you will update a student's record in the array of student records. This involves locating the correct student object by its index and modifying its properties using setter methods.

 It's time to get coding!

Use the predefined method *updateElement* method in the *Main.java* class.

- TODO 11: locate the element you need to update. Use the element's index of the array.
- TODO 12: once you have the index, access the element and update its properties. Use the 'setter' method in the *Student* class to change values like name, age, or ID.

```
public class Main{

    public static void updateElement(int indexToUpdate, Student[]originalArray){

        // TODO 11: locate the element you need to update. Use the element's index
        of the array.

        //TODO 12: once you have the index, access the element and update

    }

}
```

Let's walk through the *printArray* method, which is provided in the *Main.java* class. This method is designed to print the details of all students in the array. It's straightforward yet crucial for observing the current state of each array element.

In what scenario might this be relevant? Well, imagine you need to present the current list of students to the university administration or for a report. Displaying all student records clearly and accurately is essential for transparency and effective communication. Let's dive in.

 It's time to get coding!

Use the predefined method *printArray* method in the *Main.java* class.

- TODO 13: include a print statement to indicate the current action being performed, such as 'Removing student', 'Adding student', or 'Updating student'.
- TODO 14: use a for loop to go through each element in the array.
- TODO 15: within the loop, use the getter method to fetch each student's details and print them using the student object.

```
public class Main{
```

```

public static void printArray(String message, Student[] students){

    //TODO 13:include a print statement to indicate the current action being
    performed

    //TODO 14: use a 'for loop' to go through each element in the array

    // TODOs continue

}

}

```

Imagine you are nearing the deadline for the student record management system project, and it is crucial to ensure that all functionalities work as expected before the final deployment. Testing the code thoroughly will help you identify and fix any issues, ensuring a smooth and error-free operation of the system.

In this task, you will compile and run your code to verify that all operations on the student array (adding, removing, updating, and displaying) function correctly. You will compare the output of your program to the expected results to ensure accuracy.

- Compile and run the code.
- Compare the output of your program to the expected results.

Expected Output

```
Remove element at 4
```

```
Operation:Remove
```

```
Student Name : John
```

```
Student Name : Mary
```

```
Student Name : Krish
```

```
Student Name : Sara
```

```
Add new student rollNumber :6, Name : shally, Age: 3
```

```
Operation:Add
```

```
Student Name : John
```

```
Student Name : Mary
```

Student Name : Krish

Student Name : Sara

Student Name : Harry

Student Name : shally

Update an element at index 2

Operation:Update

Student Name : John

Student Name : Mary

Student Name : Jenny

Student Name : Sara

Student Name : Harry

Congratulations! You've completed the *Activity: Manipulating arrays*. In this lab, you've manipulated arrays of objects, a fundamental skill for any software developer. Your goal was to modify the *Main.java* class and remove, add, update, and display elements in an array.

By working through these tasks, you've developed a solid understanding of array manipulation and prepared yourself for more complex real-life coding tasks. Being able to manipulate arrays of objects is crucial in contexts such as inventory management, where you track and update product details and stock levels, and customer relationship management (CRM), where maintaining accurate customer information and transaction histories is essential for providing personalized services. This foundational knowledge positions you well for tackling diverse data management challenges in real-world applications.

