The following topics cover Java features. Choose **one topic from Group 1** and **one topic from Group 2** for your initial post. For each of the topics you have selected, provide code examples to elaborate on your thoughts.

**Group 1**

* Enums
* Wrapper Classes
* Regular Expressions

**Group 2**

* Comparable
* Comparator

From the first group, I choose to elaborate on enums. Enums are a constant variable in Java, meaning the values do not change. An enum is a keyword that can implement interfaces. An enum must be declared, and it can exist inside a class. You can loop through an enum or use it with a switch statement. An enum can have attributes and methods, but cannot be overridden. Programiz (n.d) states that enums are typically displayed in uppercase letters.

Here's an example of the Java Enum Class (Programiz, n.d.):

enum Size {

SMALL, MEDIUM, LARGE, EXTRALARGE

}

class Test {

Size pizzaSize;

public Test(Size pizzaSize) {

this.pizzaSize = pizzaSize;

}

public void orderPizza() {

switch(pizzaSize) {

case SMALL:

System.out.println("I ordered a small size pizza.");

break;

case MEDIUM:

System.out.println("I ordered a medium size pizza.");

break;

default:

System.out.println("I don't know which one to order.");

break;

}

}

}

class Main {

public static void main(String[] args) {

Test t1 = new Test(Size.MEDIUM);

t1.orderPizza();

}

}

From the second group, I will focus on the comparator. In Java, the comparator is an interface. You can use the comparator interface by importing the java.util.package. The comparator interface is great to use when sorting multiple things within a class and when separating sorting logic from a class (GeeksforGeeks, 2025). An example of when the comparator interface would be helpful is when tracking a customer's name, order number, account number, addresses, and invoice price. The sort() method can be utilized to sort elements. An even better way to use a comparator is with a lambda expression and return with a compare method().

Here's an example of a comparator with a lambda expression (GeeksforGeeks, 2025).

// Alternative Method

import java.util.\*;

// Define the Student class

class Student {

String name;

Integer age;

// Constructor

Student(String name, Integer age) {

this.name = name;

this.age = age;

}

public String getName() {

return name;

}

public Integer getAge() {

return age;

}

// Method to print student details

@Override

public String toString() {

return name + " : " + age;

}

}

public class ComparatorHelperClassExample {

public static void main(String[] args) {

List<Student> students = new ArrayList<>();

students.add(new Student("Ajay", 27));

students.add(new Student("Sneha", 23));

students.add(new Student("Simran", 37));

students.add(new Student("Ankit", 22));

students.add(new Student("Anshul", 29));

students.add(new Student("Sneha", 22));

// Original List

System.out.println("Original List:");

// Iterating List

for (Student it : students) {

System.out.println(it);

}

System.out.println();

// Sort students by name, then by age

students.sort(Comparator.comparing(Student::getName).thenComparing(Student::getAge));

// Display message after sorting

System.out.println("After Sorting:");

// Iterating using enhanced for-loop after sorting ArrayList

for (Student it : students) {

System.out.println(it);

}

}

}

**References**

GeeksforGeeks. (2025, April 16). *Java Comparator Interface*. GeeksforGeeks. [https://www.geeksforgeeks.org/java-comparator-interface/#](https://www.geeksforgeeks.org/java-comparator-interface/)

Programiz. (n.d.). *Java enum & enum Class (With Examples)*. Www.programiz.com. Retrieved April 14, 2025, from <https://www.programiz.com/java-programming/enums>

**Assignment Requirements and Grading:**

* + 1. An initial post of approximately 250 words is due by **Thursday, 11:59 p.m., CST**.
    2. For the initial post to be considered substantive, it should be at least 250 words in length and fully cover the topics being presented. Single sentence definitions or responses will not be awarded points.
    3. Submit your post by clicking on the assignment link above, then Create Thread. You must create a thread in order to view your peers' posts. Tip: Create your post in a Word document and then copy and paste your work into the thread.
    4. A minimum of three (3) responses to the original threads of other students, of 100-200 words each, are due by **Sunday, 11:59 p.m., CST**.
    5. To view the rubric grading criteria, click on the following link: [Discussion Board Grading Rubric.](https://content.bellevue.edu/cst/csd/rubricdbv3.pdf)

Hey, Arely! Your discussion post for this week is very nicely said. The source code for the Comparable and Comparator does a great job of adding to your explanation of those interfaces in Java. Having a refresher on the comparator and comparable interface was immensely helpful when completing the program for this week’s module. We learn so much information every week that it can be challenging to remember every bit, so it’s nice when we get refreshers like this and expand on them in the process. The one thing I will say is that I think we were supposed to choose a topic from each group, not just from one group. But otherwise, great job!

Hello, Samir. You did a nice job on your post for this module. I also chose to write around enums for group 1. Something I found interesting about enums is that usually the preferred way to use them is by writing them in all caps. This distinction can make it easier to recognize immediately that an enum is set, along with the enum keyword being included. Your code example that you included accurately shows enums in action. Pairing them with switches can be a good way to implement them in code. Did you do something similar to the code you included for your comparable interface example in your module 6 program assignment this week?

Hey, Nima. I think your post demonstrates a strong understanding of enums and the Comparable interface in Java. You included code for both, add to your explanations, so they are a great pairing. I need to utilize them more often when I know I am working with constants because they really help to simplify code. I see how enums are helpful for many scenarios, especially when shifting through settings. I am glad you mentioned how enums work great with switch statements. This can be a benefit when coding. I found the compare to method to be a great feature when sorting through an array.