**Julio Pochet Edmead**  
**Module 6 Discussion Post – Wrapper Classes & Comparator**

This week I focused on two Java features that I’ve seen before but never really appreciated until now: **Wrapper Classes** and the **Comparator** interface.

Starting with wrapper classes—Java doesn’t allow you to directly use primitive types like int or double in collections such as ArrayList. Instead, you use their object versions like Integer, Double, and so on. What’s cool is Java handles the conversion for you automatically using autoboxing and unboxing.

Here’s a quick example:

int num = 10;

Integer wrapped = num; // autoboxing

int unwrapped = wrapped; // unboxing

You’ll run into this a lot, especially when doing something like:

ArrayList<Integer> scores = new ArrayList<>();

scores.add(95);

scores.add(88);

Without wrapper classes, you couldn’t even store these numbers in the list.

From Group 2, I went with **Comparator** because it gives more flexibility for sorting compared to Comparable. You don’t have to modify the class you’re sorting, you just write the comparison logic separately.

Here’s an example that sorts a list of strings by length:

List<String> colors = Arrays.asList("yellow", "red", "blue");

colors.sort((a, b) -> Integer.compare(a.length(), b.length()));

System.out.println(colors); // Output: [red, blue, yellow]

This makes it really easy to sort data however you want—by name length, price, date, or anything else. It’s especially helpful in projects where you might need to sort the same objects in different ways without rewriting your class.

These two features, **wrapper classes and comparators**, are small things that make Java a lot more powerful and easier to work with.