

Class

9

CSC 240 Computer Science III

Chapter 10 Synthesis & Mega Evolution Capstone

Dr. Si Chen (schen@wcupa.edu)



Tue / Thu · 9:30 – 10:45 AM · Anderson Hall 315
Fall 2026 · Week 5 · Thursday, September 24, 2026
wcu.ninja/csc240

Warm-up: abstract classes & interfaces

- Quick, out loud - why couldn't you write `new Shape()`; last class, once Shape was abstract?
- Q1: a class implements both Resizable and MegaEvolvable. How many classes can it extend at the same time?
- Q2: an interface method has no body. What's the one thing every implementing class MUST provide?
- Q3: what's the difference between a capability (interface) and a family (abstract class extends)?

TODAY

Two Methods You've Always Had

Every `toString()` you've written this semester quietly overrides a method from `Object`. Today: name that rule, fix `Object`'s broken `equals()`, and add one more interface - `Comparable` - so your own classes can sort themselves.

Object's defaults are rarely what you want

```
Circle a = new Circle(3);
Circle b = new Circle(3);

System.out.println(a);
// Circle@1b6d3586 (Object default)

System.out.println(a == b);
// false

System.out.println(a.equals(b));
// false (Object default: same as ==)
```

- `toString()` default: class name + a hash code - useless for printing your own objects. You've been overriding it since the Pokedex project without being told the rule.
- `equals(Object)` default is literally `this == other` - true only for the exact same object in memory, never for two objects that just describe the same value.
- Both live on `Object`, so overriding them is the exact same mechanic as overriding `draw()` on `Shape`.

This is what you get with NO overrides.

In-class: read the fix

```
@Override
public String toString() {
    return String.format(
        "%s(area=%.2f)",
        getClass().getSimpleName(),
        area());
}

@Override
public boolean equals(Object o) {
    if (this == o) return true;
    if (!(o instanceof Circle))
        return false;
    Circle c = (Circle) o;
    return Double.compare(
        radius, c.radius) == 0;
}
```

- Predict all 5 `println()` lines in `Main.java` before running.
- Note the parameter type: `equals(Object other)`, never `equals(Circle other)` - get that wrong and you've silently overloaded instead of overridden.
- Try renaming it to `equals(Circle other)` and re-run. `@Override` would have caught this immediately - always add it.

Full code: 01_equals_and_tostring/.

Comparable<T>: sorting is just another interface

```
public abstract class Shape
    implements Comparable<Shape> {

    public abstract double area();

    @Override
    public int compareTo(Shape other) {
        return Double.compare(
            this.area(), other.area());
    }
}
```

- `Comparable<T>` is a library interface (`java.lang`) - same idea as `Resizable` or `MegaEvolvable`, just written by Oracle instead of by you.
- One required method: `compareTo()`. Negative = comes before, positive = comes after, zero = tie.
- Implement it once, and `Arrays.sort()`, `Collections.sort()` all automatically know how to order your objects - no extra code anywhere else.

Full code: 02_comparable_shapes/.

Arrays.sort() needs no extra code at all

```
Shape[] shapes = {  
    new Rectangle(4, 5),  
    new Circle(3),  
    new Triangle(6, 2),  
};  
  
Arrays.sort(shapes);  
  
for (Shape s : shapes)  
    System.out.println(s);  
// Triangle(area=6.00)  
// Rectangle(area=20.00)  
// Circle(area=28.27)
```

- `Arrays.sort()` calls each element's `compareTo()` behind the scenes - that's the entire contract it relies on.
- Try removing `implements Comparable<Shape>` from `Shape.java` and recompiling: the error names the missing capability directly.
- Same pattern you'll use again in a few minutes with `Pokemon`.

Full code: 02_comparable_shapes/.

== vs. equals()

== (identity)

- asks: are these the SAME object in memory?
- built into the language - **can't** be overridden.
- always safe and fast, but often not what you mean.
- use it for: primitives, or when you genuinely want reference identity.

equals() (your choice)

- asks: do these objects represent the SAME value?
- inherited from Object, **override** it to define “same value” yourself.
- default behavior (before you override) is identical to ==.
- use it for: comparing objects the way a person would - same radius, same Pokedex id.

In-class: finish Triangle

```
Shape[] shapes = {  
    new Rectangle(4, 5),  
    new Circle(3),  
    new Triangle(4, 3),  
    new Triangle(4, 3),  
    new Circle(3),  
};  
  
Arrays.sort(shapes);  
// ...count how many shapes  
// equal shapes[0]...
```

- Sorting already works - `compareTo()` is inherited from `Shape`, nothing to add there.
- Add `equals()` to `Triangle.java`: equal means SAME base AND SAME height, not just the same area.
- Before your fix the “equal to first” count prints 1; after it should print 2.

Starter: 03_synthesis_exercise_starter/.

Capstone: real Mega Evolution

```
public class Charizard extends Pokemon
    implements MegaEvolvable {
    private boolean megaEvolved = false;

    public void megaEvolve() {
        if (megaEvolved) return;
        attack += 46; defense += 33;
        megaEvolved = true;
    }
    // + revertMega(), isMegaEvolved(),
    //   getMegaFormName() -> "Mega Charizard X"
}
```

- Last class every Mega was hypothetical. Today: Pokemon with a REAL Mega form, using the real stats.
- `MegaEvolvable` gains one method, `getMegaFormName()`. Every implementer must now add it, and the compiler checks.
- Part 1: Charizard (TODOs 1-6). Part 2: Blastoise, with no step-by-step hints.
- HP never changes in a real Mega Evolution, so only ATK/DEF/SPD move.

Starter: 05_mega_world_starter/ (Venusaur.java is the worked example).

Send it to the Pokemon World, then Mega Evolve

```
public static Pokemon upgrade(Pokemon p) {  
    switch (p.getId()) {    // TODO 7: add 6, 9  
        case 3: return new Venusaur(...);  
        case 6: return new Charizard(...);  
        case 9: return new Blastoise(...);  
        default: return p;  
    }  
}  
  
// PokedexGUI, for every card:  
if (p instanceof MegaEvolvable) {  
    // show the "Mega Evolve!" button  
}
```

- Part 3: `upgrade()` returns a `Charizard` where the CSV row says id 6. The list is still `ArrayList<Pokemon>`. That is polymorphism.
- The GUI never checks names. It only asks `instanceof MegaEvolvable`, so the button appears the moment your class implements the interface.
- Send to World (name, ID, access code), then click Mega Evolve! Your Pokemon flashes on the big screen and turns into its real Mega sprite.
- Bonus: Charizard Y, Gyarados, Mewtwo X/Y... (stats table in README.md).

Big screen: wcu.ninja/csc240/pokemonworld

Homework 2 assigned: Payroll (Chapter 10)

- Due Tue Oct 6, 11:59 PM on D2L. 100 points, 6% of the course grade. Handout + starter: wcu.ninja/csc240.
- abstract `Employee` -> `SalariedEmployee`, `HourlyEmployee` (overtime), `Manager` extends `SalariedEmployee` (`super.weeklyPay()`).
- interface `Payable`: implemented by `Employee` AND by an unrelated `Invoice`, the same "can-do" idea as `MegaEvolvable`.
- `equals()` by id (catches a duplicate CSV row) and `Comparable<Employee>` by weekly pay: exactly today's first half.
- Start this week. Midterm (Ch. 9-10) is planned for Oct 8, and HW2 is the best practice for it.

Sources

- Gaddis, Starting Out with Java, 4th ed., Ch. 10 (inheritance, polymorphism, abstract classes, interfaces).
- Shape/Circle/Rectangle/Triangle (code/class09_synthesis/): today's primary example set for equals(), toString(), and Comparable.
- Pokemon.java: built on the CSC 240 Fall 2026 Pokedex capstone and Class 6-8 inheritance demos. Mega stats: official Pokemon X/Y base stats (via PokeAPI); sprites: PokeAPI sprite CDN.
- Java SE 17 API: Object (equals, toString, hashCode) and java.lang.Comparable.