CLASSES CHEAT SHEET

ADAMBOTS Team 245

1 A class: the blueprint

```java
// file must be named Battery.java
public class Battery {
    // fields: what it knows
    private String name;
    private double voltage;

    // constructor: runs on new
    public Battery(String name, double voltage) {
        this.name = name;
        this.voltage = voltage;
    }

    // method: what it does
    public boolean isMatchReady() {
        return voltage >= 12.0;
    }
}
```

2 Objects: the real things

```java
Battery a = new Battery("2025-A", 12.8);
// the dot: "a, do this"
boolean ok = a.isMatchReady();
```

Class name	Capital first letter: `Battery`
Variable name	lower case: `spareBattery`
`new`	builds one object from the class

3 Constructors

Same name as the class	**No** return type, not even `void`
Runs once, on `new`	Sets up the fields
`this.voltage`	"my own" field (vs. the parameter)

`voltage = voltage;` sets nothing. Write `this.voltage = voltage;`

4 References and null

Primitives	`int double boolean char`: value sits in the box
Objects	the box holds an **arrow** (a reference)
`b = a;`	copies the arrow: one object, two names
`==`	same object? `.equals()`: same contents?
`null`	an arrow to nothing

NullPointerException: you used a null variable. The message names it. Every object field needs its `new`.

5 static vs. instance

instance (normal)	one copy **per object**
`static`	one copy for the **whole class**
`static final`	constant: `kIntakeSpeed`
Static call	`Math.max(a, b)`: no object needed

"non-static ... cannot be referenced from a static context"? **Make an object.** Don't add `static`.

6 private and public

```java
private double speed;
public void setSpeed(double s) {
    speed = Math.clamp(s, -1.0, 1.0);
}
public double getSpeed() { return speed; }
```

Robot rule: fields **private**, useful methods **public**.

7 Packages and imports

```java
package frc.robot.subsystems;   // = the folder
import org.wpilib.drivers.motor.PWMSparkMax;
```

"cannot find symbol"? Typo, missing import, or out of scope. Light bulb: **Ctrl + .** (Mac: Cmd + .)

8 Many objects

```java
// 4 empty (null) slots, numbered 0 to 3
SwerveModule[] mods = new SwerveModule[4];
mods[0] = new SwerveModule(1, 2);
for (SwerveModule m : mods) { m.stop(); }

ArrayList<Battery> cart = new ArrayList<>();
cart.add(a);  cart.size();  cart.get(0);
```

Numbered from 0. Last is `size() - 1`.

9 extends

```java
public class Intake extends SubsystemBase {
    @Override public void periodic() { }
}
```

"An Intake **is a kind of** Subsystem." `@Override` replaces an inherited method.