Java differences

- Java is byte-compiled.
- Java has static types.
- Java requires us to write all declarations in a class.

Values and equality in Java

Primitive types vs objects

There are two types of values in Java: primitive values and objects.

Examples of primitive value types include int, double, boolean, and other built-in types. Java directly stores primitive values (it’s “in the box”).

Examples of object types include String, LinkedList, other library types, and user-defined classes. Java stores references to objects.

Reference equality (==) vs value equality (.equals)

Reference equality checks whether two names refer to the same object: \( x == y \).

Value equality calls a method that (typically) checks whether two potentially different objects have the same value: \( x.equals(y) \).

Java Conventions and good programming practices

These conventions and programming practices are not required or checked by Java, and not all of them are universally agreed upon by all Java programmers. For consistency, though, we’ll follow them in CS 42.

- Place field definitions at the top of the class.
- Use Javadoc (/** ... */) to document your fields and methods.
- Inside a class, always use this to refer to the members (i.e., fields and methods) of the class.
- Keep your main program separate from your class definitions.
- Usually, fields are private, and the class provides public getters and setters, if needed.
- Write tests first!
- Minimize the number of methods that access fields.
- Write a toString method for every class you implement.
- Provide (preferably through autogeneration) an equals and hashCode method for every class you implement.
- Always refer to a static field via the class. Never refer to a static field via an object.
- Provide good constructors. Explicitly initialize all fields.
class Point {
    /** the x (horizontal) coordinate */
    private double x;

    /** the y (vertical) coordinate */
    private double y;

    public Point(double x, double y) {
        this.x = x;
        this.y = y;
    }

    public double getX() {
        return this.x;
    }

    public void setX(double x) {
        this.x = x;
    }

    public double getY() {
        return this.y;
    }

    public void setY(double y) {
        this.y = y;
    }

    /**
     * returns the sum of this point and another
     * @param other another Point object
     * @return a new Point, the sum of this and other
     */
    public Point add(Point other) {
        return new Point(this.getX() + other.getX(),
                         this.getY() + other.getY());
    }
}