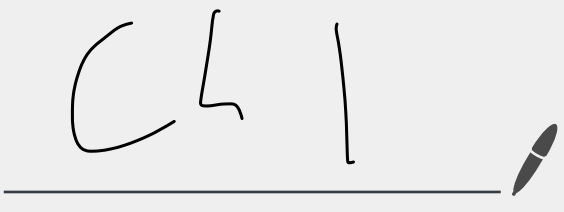
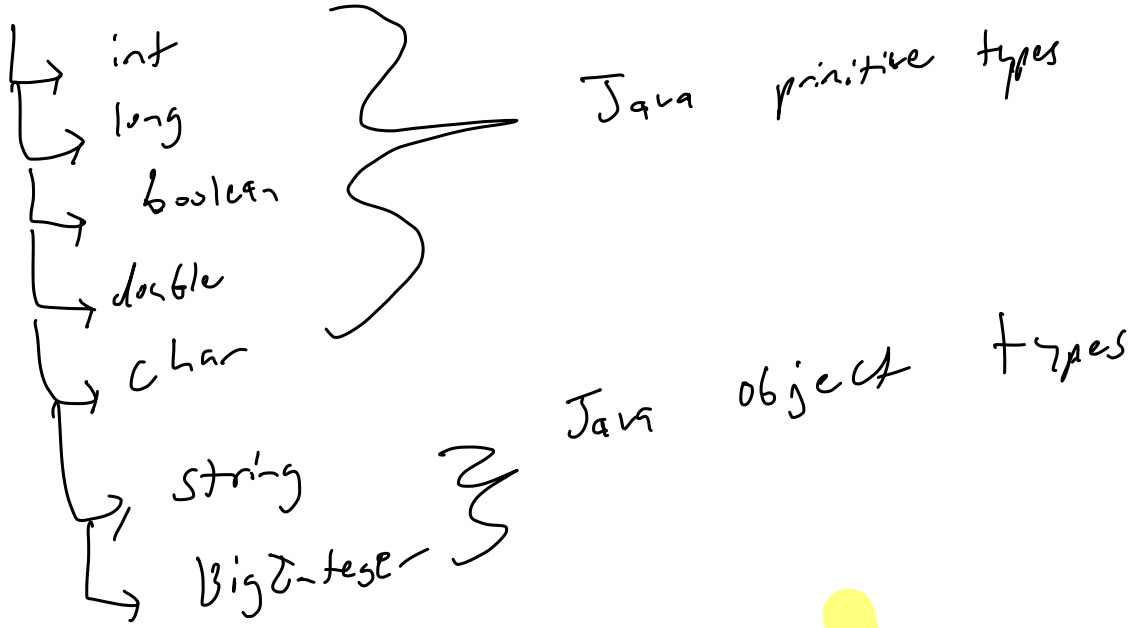


Types

- set of values, along with operations on those values



Static Typing/Checking

- types are known at compile time
- bugs are checked for at compile time

Dynamic Typing/Checking

- types are inferred at runtime
- bugs are checked during execution

Static checking can catch:

- syntax errors, like extra punctuation or spurious words. Even dynamically-typed languages like Python do this kind of static checking. If you have an indentation error in your Python program, you'll find out before the program starts running.
- wrong names, like `Math.sine(2)`. (The right name is `sin`.)
- wrong number of arguments, like `Math.sin(30, 20)`.
- wrong argument types, like `Math.sin("30")`.
- wrong return types, like `return "30";` from a function that's declared to return an `int`.

Dynamic checking can catch:

- illegal argument values. For example, the integer expression `x/y` is only erroneous when `y` is actually zero; otherwise it works. So in this expression, divide-by-zero is not a static error, but a dynamic error.
- unrepresentable return values, i.e., when the specific return value can't be represented in the type.
- out-of-range indexes, e.g., using a negative or too-large index on a string.
- calling a method on a `null` object reference (`null` is like Python `None`).

Immutability

- to make a reference immutable in Java, use:

final

Summary

The main idea we introduced today is **static checking** . Here's how this idea relates to the goals of the course:

- **Safe from bugs.** Static checking helps with safety by catching type errors and other bugs before runtime.
- **Easy to understand.** It helps with understanding, because types are explicitly stated in the code.
- **Ready for change.** Static checking makes it easier to change your code by identifying other places that need to change in tandem. For example, when you change the name or type of a variable, the compiler immediately displays errors at all the places where that variable is used, reminding you to update them as well.