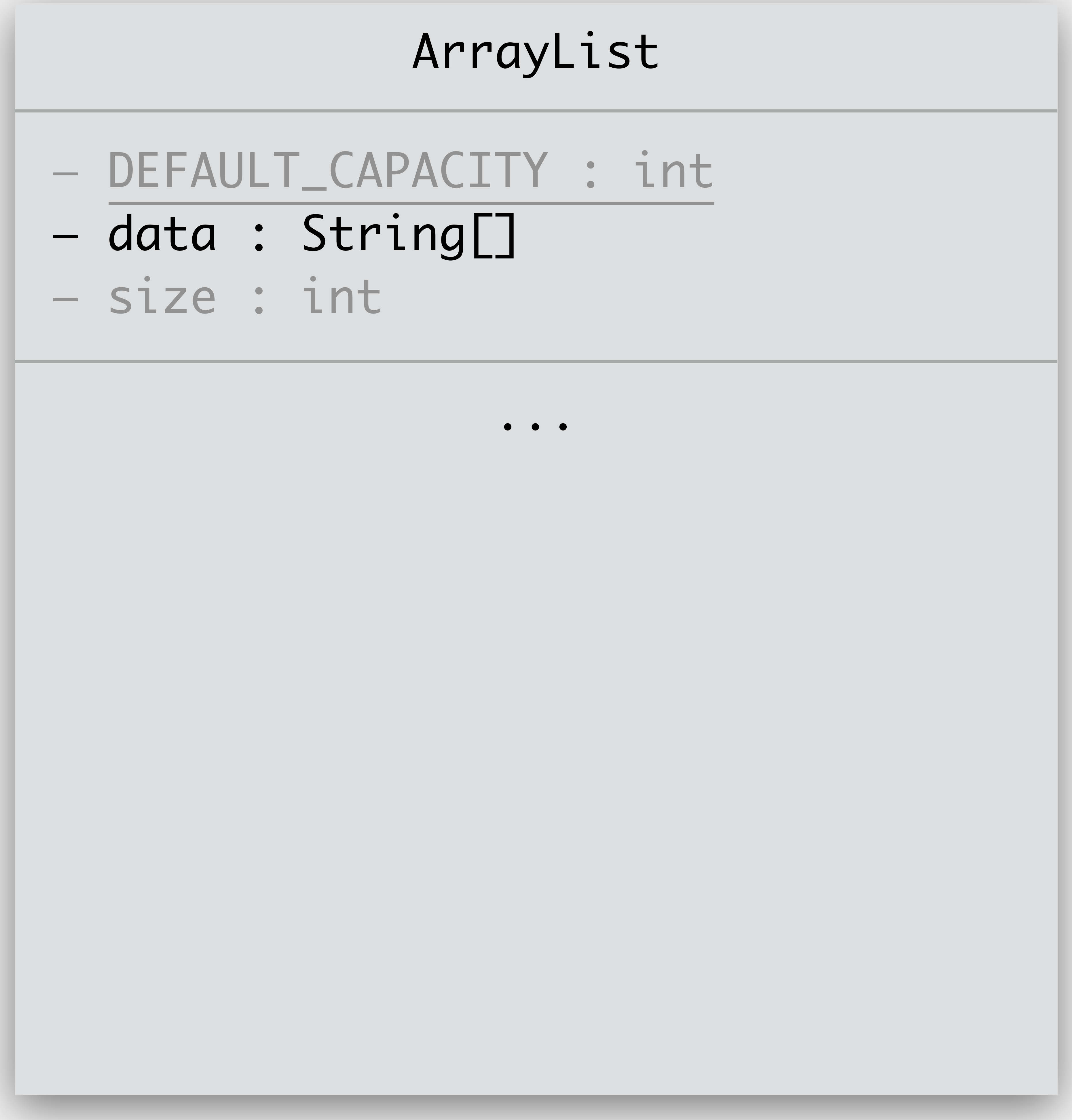
UNIVERSITY *of* WISCONSIN
LA CROSSE
COMPUTER SCIENCE

Week 06 (Part 2): Generics

CS 220: Software Design II — D. Mathias

Array List Revisited



ArrayList

- DEFAULT_CAPACITY : int
- data : String[]
- size : int

...

Our implementation isn't flexible

- can only hold string values

How can we create a data structure that holds *any* object type?

- option 1: use `Object` type and cast
- option 2: *generics*

Option 1: Use Object Type and Cast

```
Object obj1 = new Professor("Mathias", “David");
Object obj2 = "some string";

// this is a hassle
String str2 = (String) obj2;

// this causes a ClassCastException
String str2 = (String) obj1;
```

`Object` is the class at the top of the Java class hierarchy

- *everything* descends from it

Problems

- clunky to cast frequently
- potential exceptions at runtime

Option 2: Generics

generics allow a class to use placeholder values for different variable types that the programmer will specify during variable declaration

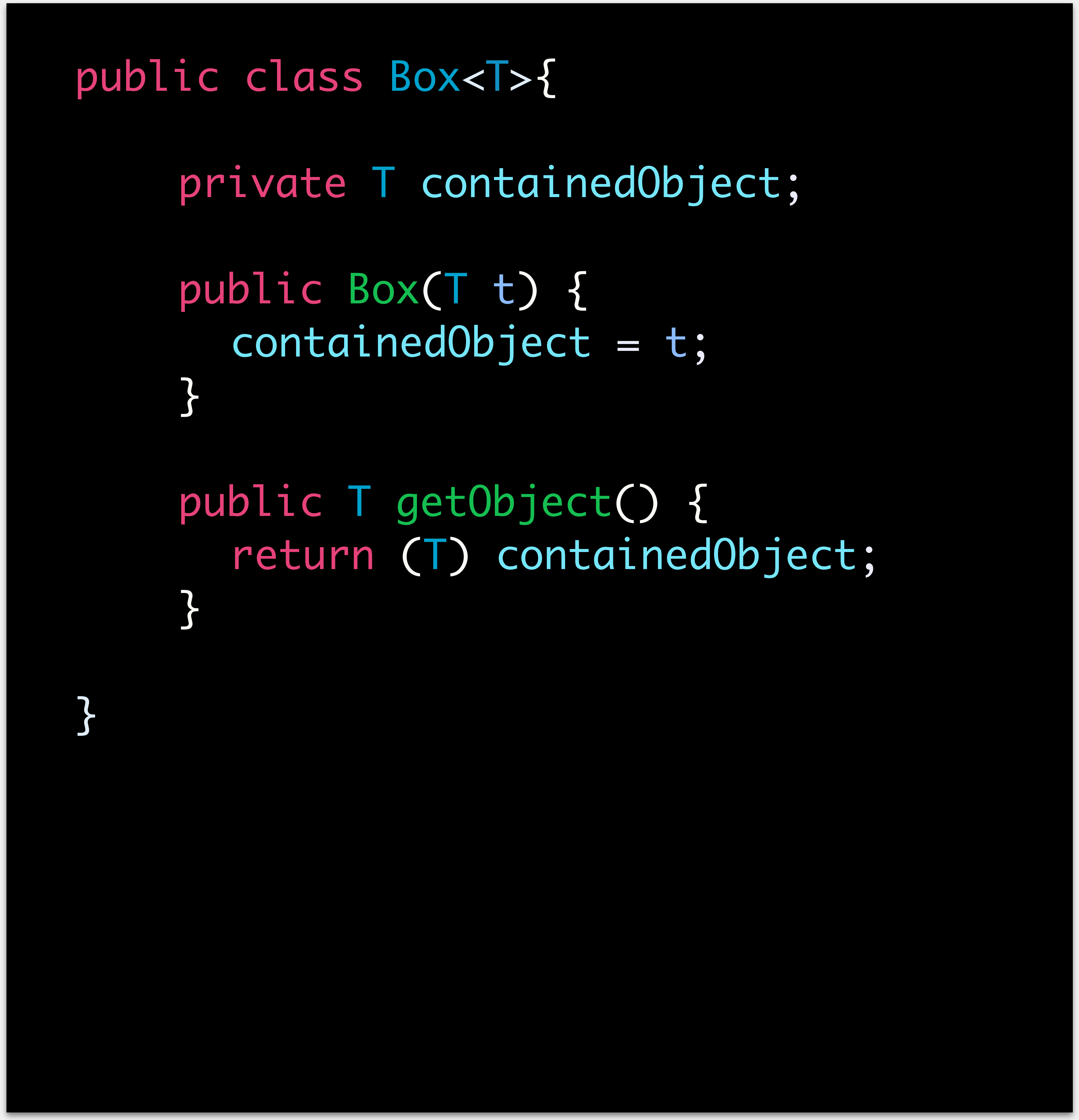
- no more casting!

Introduced in Java 5

Ensures types can be checked at compile time, rather than runtime

- i.e., prevents `ClassCastExceptions`

Generics Basics



```java
public class Box<T>{

    private T containedObject;

    public Box(T t) {
      containedObject = t;
    }

    public T getObject() {
      return (T) containedObject;
    }

}
```

Placed after the class identifier in the signature in angle brackets

- multiple generic variables? comma separated

Can only represent class types

- no primitives!
- more on this later…

Generics Basics

```java
public class Box<T>{

    private T containedObject;

    public Box(T t) {
      containedObject = t;
    }

    public T getObject() {
      return (T) containedObject;
    }

}
```

Named with a single capital letter

- visually different than an actual class type
- convention

Common conventions (not required):

- E for an element (used lots by `Collection`)
- K for key
- V for value
- N for number
- T for type
- S, U, V for additional types

Wrapper Classes

Cannot use primitive types as generic types

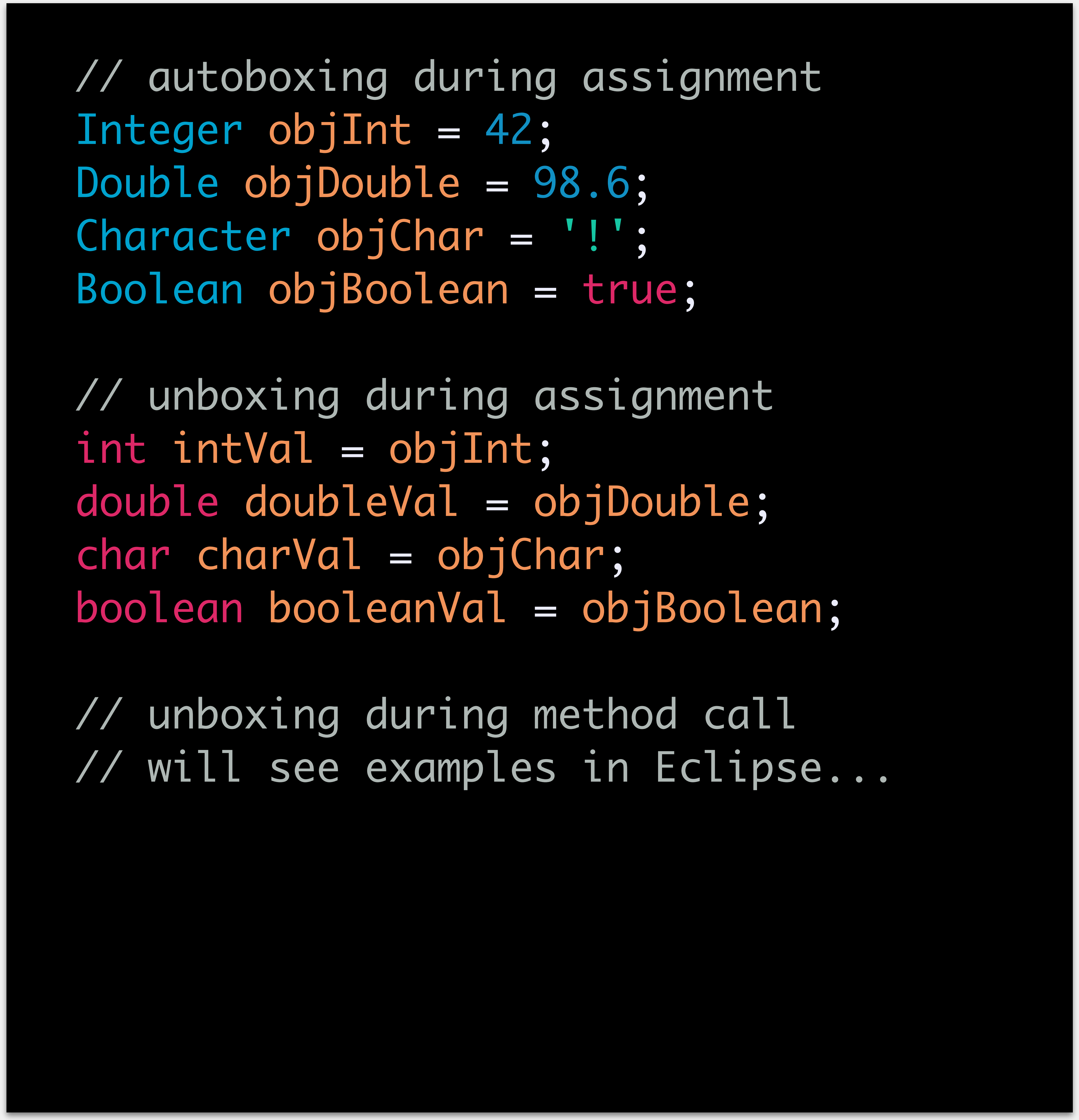
wrapper classes are class types that correspond to and hold primitive values

- introduced to make primitives work with generics

Can convert between primitive type and wrapper class

primitive type	wrapper class
`byte`	`Byte`
`short`	`Short`
`int`	`Integer`
`long`	`Long`
`float`	`Float`
`double`	`Double`
`char`	`Character`
`boolean`	`Boolean`

Autoboxing and Unboxing



```
// autoboxing during assignment
Integer objInt = 42;
Double objDouble = 98.6;
Character objChar = '!';
Boolean objBoolean = true;

// unboxing during assignment
int intVal = objInt;
double doubleVal = objDouble;
char charVal = objChar;
boolean booleanVal = objBoolean;

// unboxing during method call
// will see examples in Eclipse...
```

Rarely use wrapper class constructor to convert a primitive value to an object

autoboxing automatically converts a primitive value to its wrapper class type

unboxing automatically converts the primitive value stored in a wrapper object to its primitive type

This is done during assignment or when used as arguments to corresponding types

Generics and Arrays

Cannot have arrays of generic types

- why? it's complicated…ask in CS 421 & CS 442

How can we implement a generic array in `ArrayList`?

- answer: have the array hold type `Object`, `ArrayList` will take care of casting

Exercise: Pair Class

Write a class called `Pair` that does the following:

- holds two values of (possibly) different types (use generics)
- constructor should take in these two values and store them in global variables
- two `get` methods for getting the different parts of the pair
    - feel free to name these as you see fit

This class could be used to hold…

- …an x, y coordinate
- …a student object and their grade
- …many different things!