Classes as Types & Reference Variables

What are Data Types?

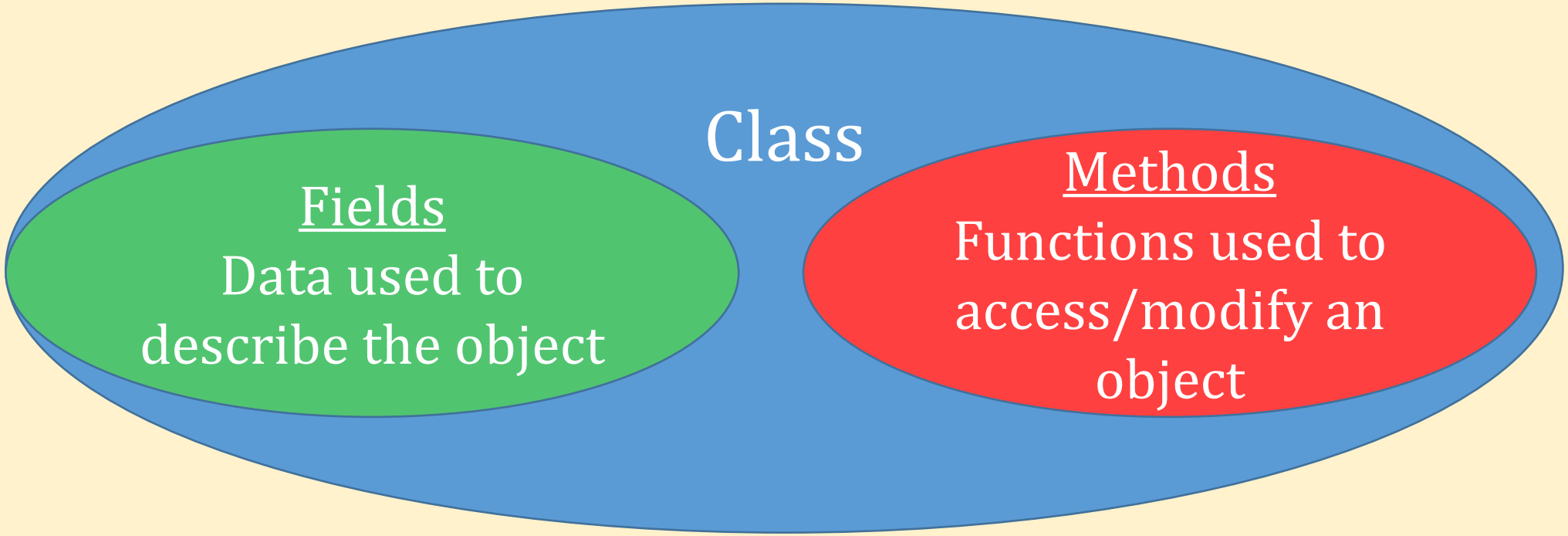
Chap 4

- A data type describes how many bits make up the data
- A data type describes how bits should be interpreted
- A data type describes what operations are allowed

Type	Bits	Size	Encoding	Value
int	0000 0000 0000 0000 0000 0000 0000 0110	32	$\sum_{i=0}^{30} b_i \times 2^i$	6
char	0000 0000 0100 0111	16	UTF-16	‘G’
float	0100 0000 0100 1000 1111 0101 1100 0011	32	$-1^S \times 1.sig \times 2^{exp}$	3.14

What are Classes?

- A class describes how many fields make up the data
- A class describes how fields should be interpreted
- A class describes what operations are allowed



Classes as Types

Sect. 2.8

- Primitive types
  - pre-defined types known to Java
  - only 8 : boolean, byte, short, int, long, char, float, double
- Java allows "types" which are ***references*** to objects
  - "Type" is the class name
  - Result is a reference variable… like a pointer in C, but with extra rules
  - Reference variables can be thought of as a "handle" on the encapsulated object
- Distinguish between the two because classes are capitalized

Comparing Types and Classes as Types

Data Types	Classes as Types
• A data type describes how many bits make up the data • A data type describes how bits should be interpreted • A data type describes what operations are allowed	• A class describes which fields make up the object • A class describes how to interpret its fields • A class defines what methods may be applied to an object

Reference Variables as Handles

- The handle only fits objects in a specific class
- We need the handle to be able to pass the object around and/or use that object
- The handle is either attached to a specific object or not attached to any object (null)

- The handle is attached to an object with an assignment statement

Declaring Reference Fields Syntax

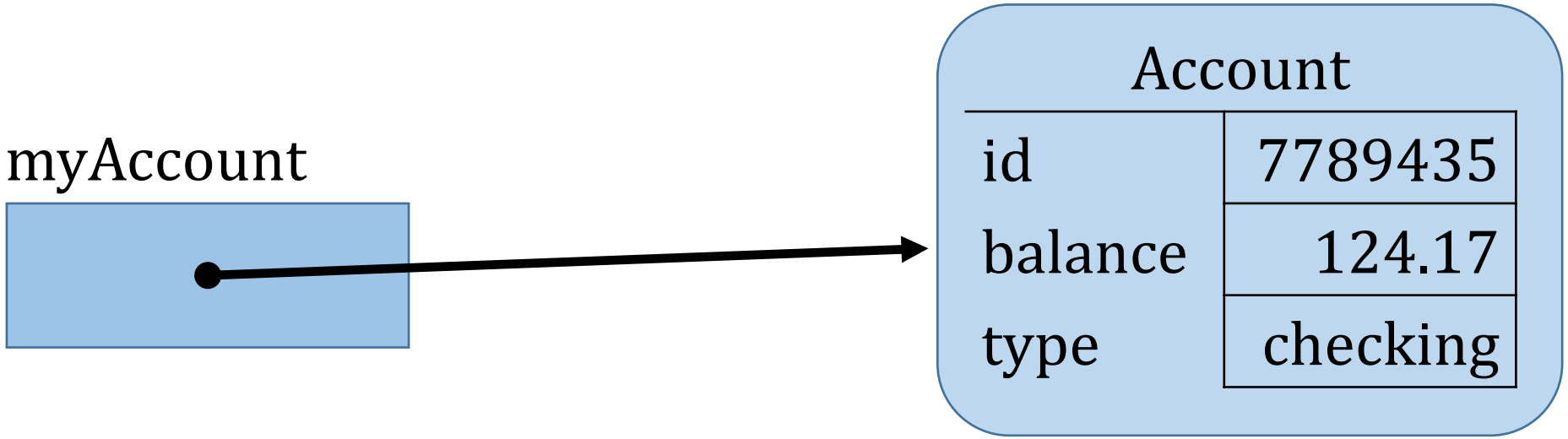
```
class classname {
        class_name reference_name;
```

or

```
        class_name reference_name= reference;
}
```

Reference Variables Schematically

```
Account myAccount = new Account(124.17);
```



Example Reference Field Declarations

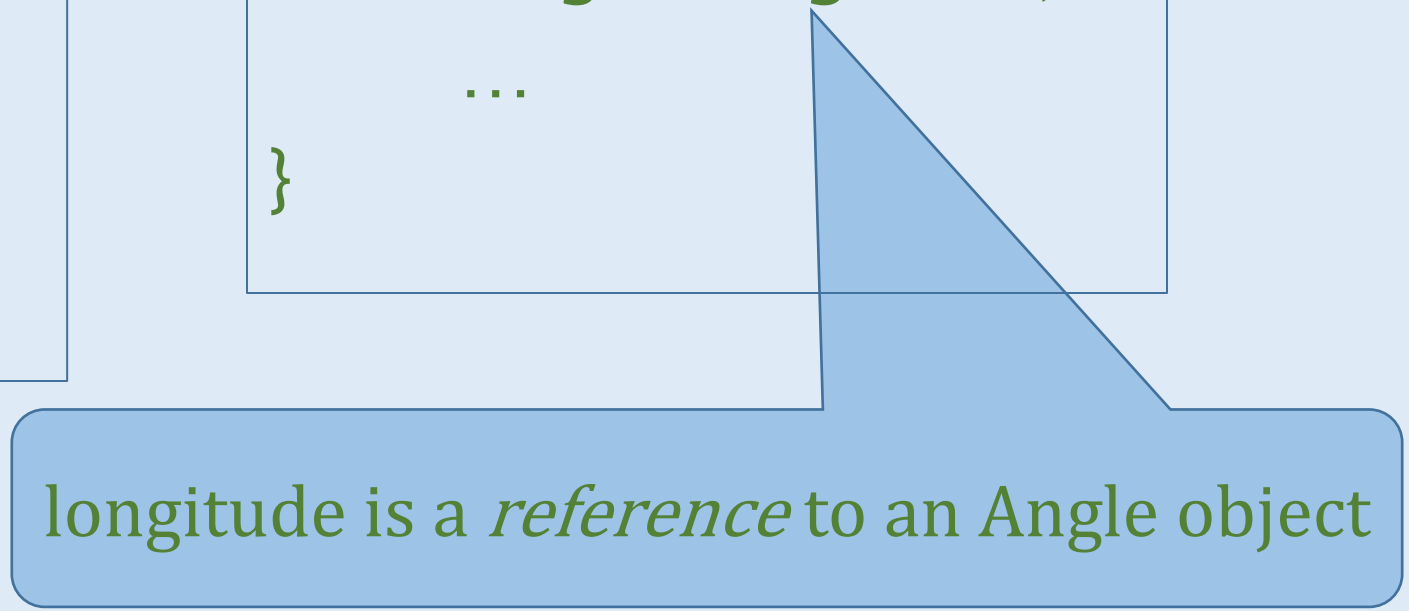
```
class Car {
        String make;
        String model;
        short year;
        Color color;
        FullName owner;
        Position location;
        double mileage;
        …
}
```

- How Cars are Described
  - make: “Porsche”
  - model: “911”
  - year: 1973
  - color: Grey
  - owner: I wish
  - location: → long/lat
  - mileage: 3,289,042.7

Field Declarations with References

```
class Angle {
        short degrees;
        byte minutes;
        double seconds;
        char nsew;
}
```

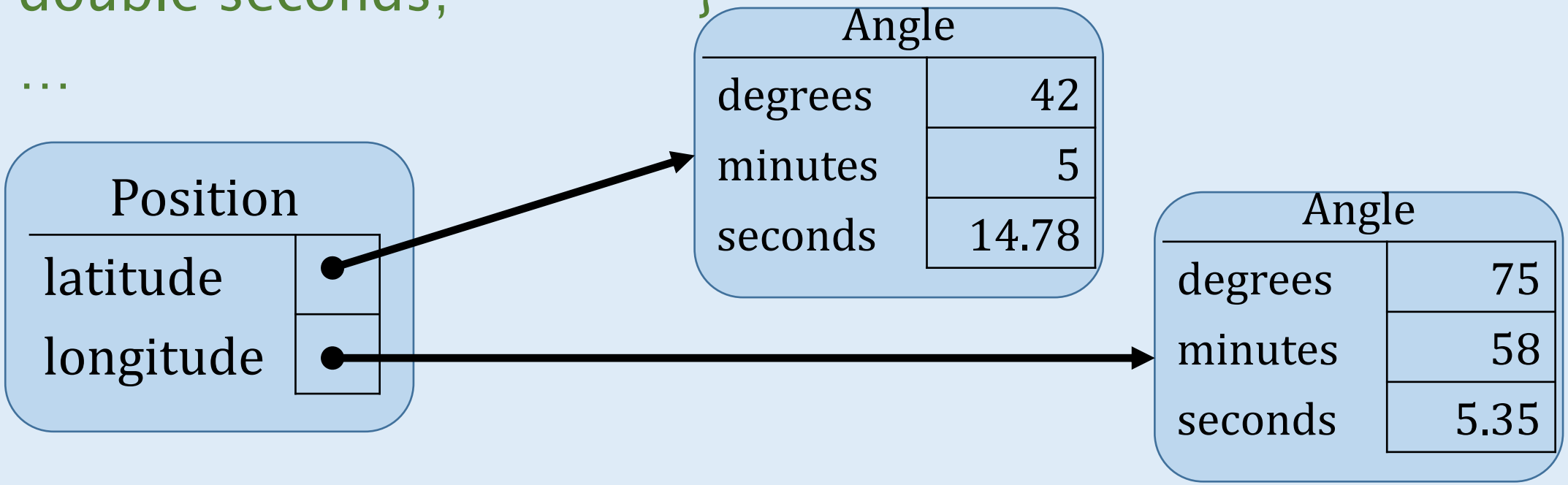
```
class Position {
        Angle latitude;
        Angle longitude;
        …
}
```



Field Declarations with References

```
class Angle {
        short degrees;
        byte minutes;
        double seconds;
        …
}
```

```
class Position {
        Angle latitude;
        Angle longitude;        …
}
```



Instantiating a Reference Variable

- ***Reference Variable***
  - A variable which references an object in a specific class
  - We need to connect the variable to a specific instance of an object
- The connection process is called "instantiating" the variable
  - Fancy word for assigning an object to the variable
- Before we assign something, the variable is "uninstantiated"
  - Fancy word that means "doesn't reference any object yet"
- Reference Variables need to be ***instantiated*** before they are used

Instantiating Reference Variables

- Using "new" keyword and class ***constructor*** to create a new object

```
myCounter = new Counter();
myAccount = new Account(124.17);
String str= "My String";
//shortcut for:  String str = new String("My String");
output = new java.io.PrintWriter(
              new java.io.FileOutputStream("out.txt"));
java.awt.Rectangle rect = new
       java.awt.Rectangle(10,30,25,60);
```

Combined Declaration & Instantiation

- Most typically, java code declares a reference variable, and instantiates that reference variable with a new object in the same statement

```
Counter myCounter = new Counter();
```

- Notice the required redundancy required
  - First “Counter” is the type of variable “myCounter”
  - Second “Counter” is the invocation of the Counter class creation method

Reference Variable Usage

- Reference variables may be declared and instantiated
- Reference variables may be re-assigned
  - reference a different object in the same (or related) class or null
- Reference variables may be queried to determine information about the object being referenced ("reflection")
- No other operations are allowed on reference variables!
  - No arithmetic allowed!

Instantiating Library Reference Variables

- To reference the PrintWriter method in the io sub-package of the java package, we need to specify java.io.PrintWriter
- Writing package names every time is a nuisance

```
java.io.PrintWriter output;
output = new java.io.PrintWriter( new java.io.FileOutputStream("out.txt"));

java.awt.Rectangle rect = new  java.awt.Rectangle(10,30,25,60);
```

Importing Packages

- importing a package allows you to use components without package names

```
import java.io.*;
import java.awt.Rectangle;
class Xyz {
        printWriter output;
        …
  output = new PrintWriter( new FileOutputStream("out.txt"));
  Rectangle rect = new Rectangle(10,30,25,60);
}
```