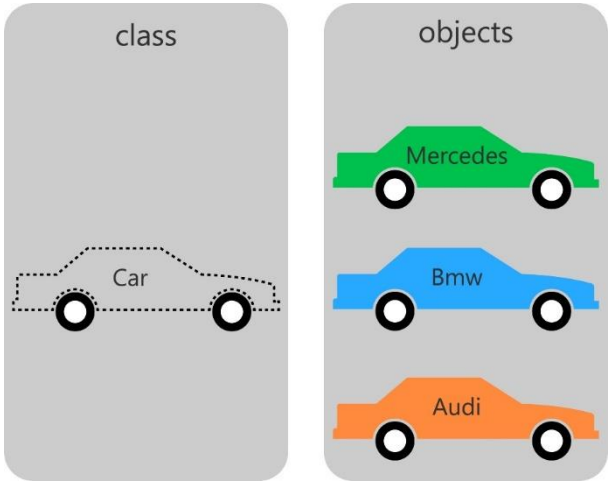
Data Types in Java

Example Class: Car

How Cars are Described

- Make
- Model
- Year
- Color
- Owner
- Location
- Mileage



Actions that can be applied to cars

- Create a new car
- Transfer ownership
- Move to a new location
- Repaint
- Delete a car

What's in a class?

- Fields – The data used to describe an object in the class

- Methods – The functions used to manipulate an object in the class

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

Java Primitive Data Types

Sect. 4.1.1

Flavor	Type	Bits	Range	Precision
logic	boolean	? (8)	true or false	exact
Integer	byte	8	-128 to +127	
	short	16	-32,768 to +32,767	
	char	16	0 to 65,535	
	int	32	$\sim -2.14 \times 10^9$ to $\sim 2.14 \times 10^9$	
	long	64	$\sim -9.22 \times 10^{18}$ to $\sim 9.22 \times 10^{18}$	
floating point	float	32	$\sim -10^{38}$ to $\sim +10^{38}$	~7 digits
	double	64	$\sim -10^{308}$ to $+10^{308}$	~15 digits

Declaring Local Variables

- Syntax (within a method)

```
type name;
```

or

```
type name=constant;
```

- Examples

```
int j;
int count=0;
```

count is an instance of an integer whose initial value is zero

Declaring Variables

Sect. 2.2

- Variables must be ***declared*** and ***initialized*** before they are used
  - Java is an imperative language, and this is typical
  - Enables compiler to check to make sure variables are used correctly
  - Declaration and initialization can be combined

- Declaration
  - specifies "type" – how this variable will be used and what values are valid
  - specifies "name" – how you will refer to this value from now on
- Initialization
  - specifies the first value that the variable will have.

Examples of primitive variable declaration

```
int value =  1; // declaration and initialization

int input; // declaration
input = 5; // initial value or a change of value

char letter = 'g';
char aKoreanChar = '\ud55c';    //  한
```

More examples:

```
long large = 12345678901234567890L;
boolean choice = false;

double  precipAug = 3.59;
float  precipJuly = 2.69F;

byte  tiny  = (byte)110;   // values from -128 to 127
short  small = (short) 25000;  // from -32768 to 32767
```

Declaring Fields (or Instance Variables)

- For each data that is used to define an object
  - We need to define the name of the field
  - We need to define the type of the field

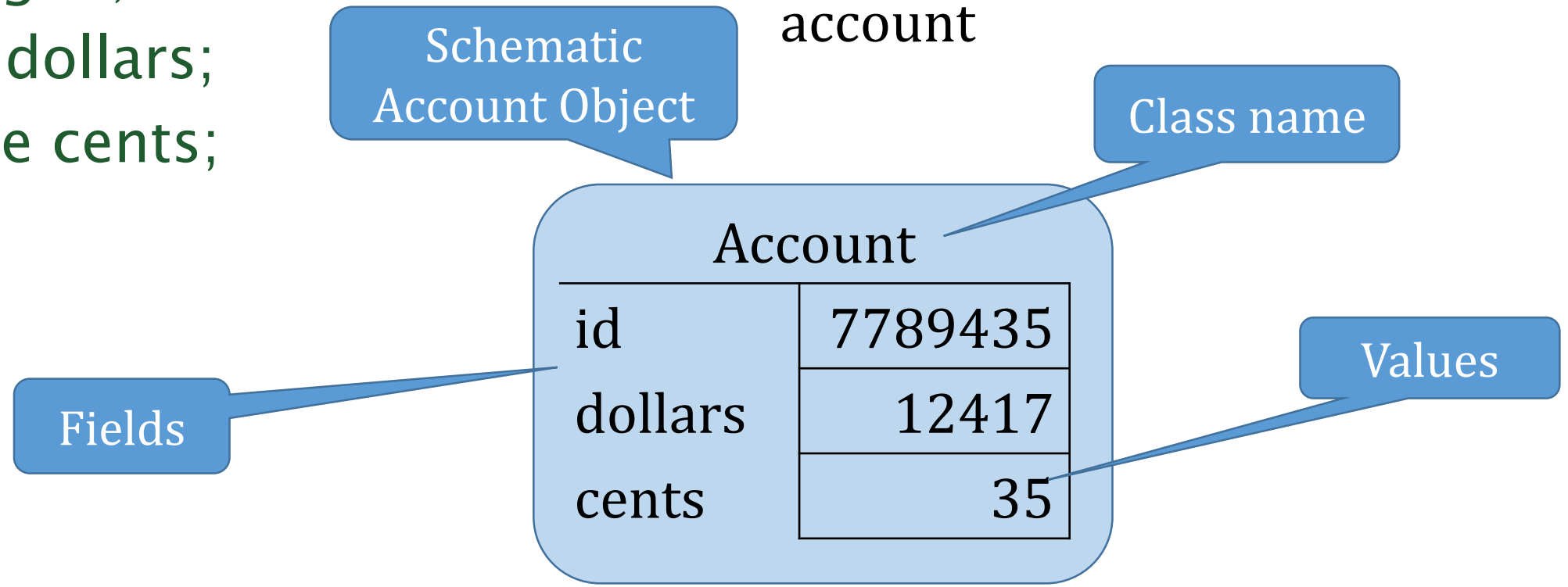
- Field Declaration Syntax:

```
class classname {
        type fieldname;
        type fieldname;
        …
}
```

Simple example with Primitive Fields

```
class Account {
   long id;
   int dollars;
   byte cents;
   …
}
```

- Fields used to define what we want to model about an account



Field Declarations with References

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

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

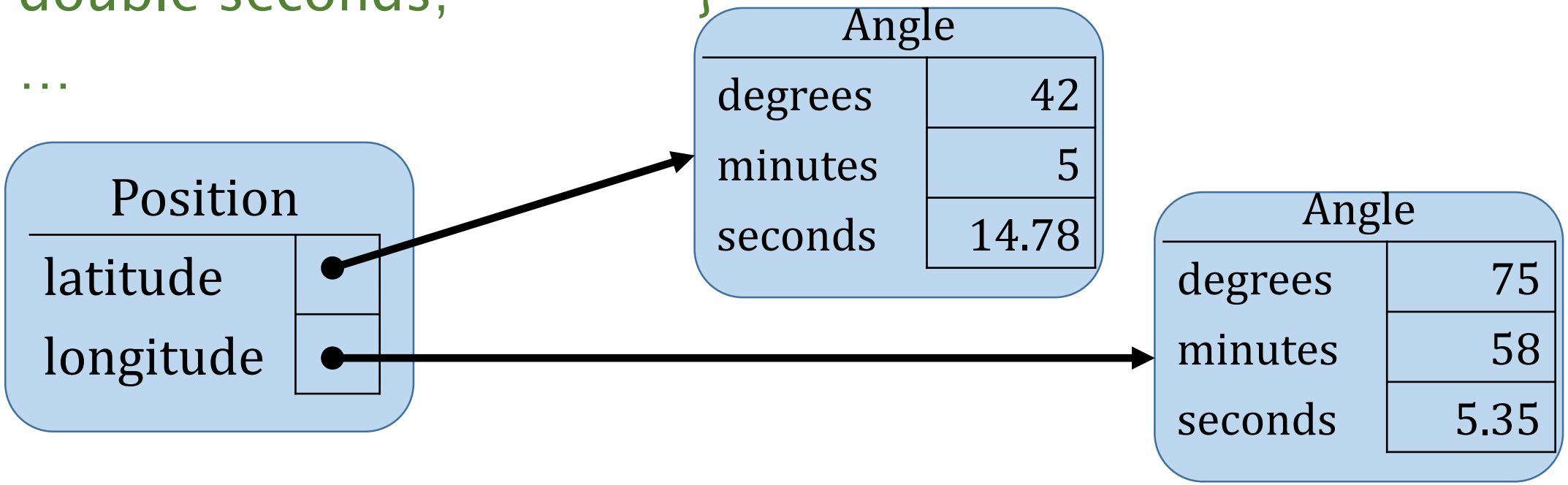
seconds is an instance of a double value

longitude is a *reference* to an Angle object

Field Declarations with References

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

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



Classes as Types

- 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

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 how much data make up the object • A class describes how to interpret its fields • A class defines what methods may be applied to an object

Example Field Declarations

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

- How Cars are Described
  - Make
  - Model
  - Year
  - Color
  - Owner
  - Location
  - Mileage

Local Reference Variable Examples

- We can declare reference variables:

```
Counter myCounter;
Account myAcc;
String str;
java.io.PrintWriter output;
java.awt.Rectangle rect;
```

Chap. 3

- Reference Variables need to be ***instantiated*** before they are used
  - Instantiation defines which object the reference variable refers to

Instantiating Reference Variables

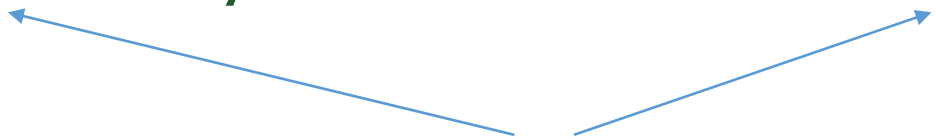
- Need to call class ***constructor*** to create a new object

```
myCounter = new Counter();
myAcc = new Account(150.0);
String str= "My String";
//same as:  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

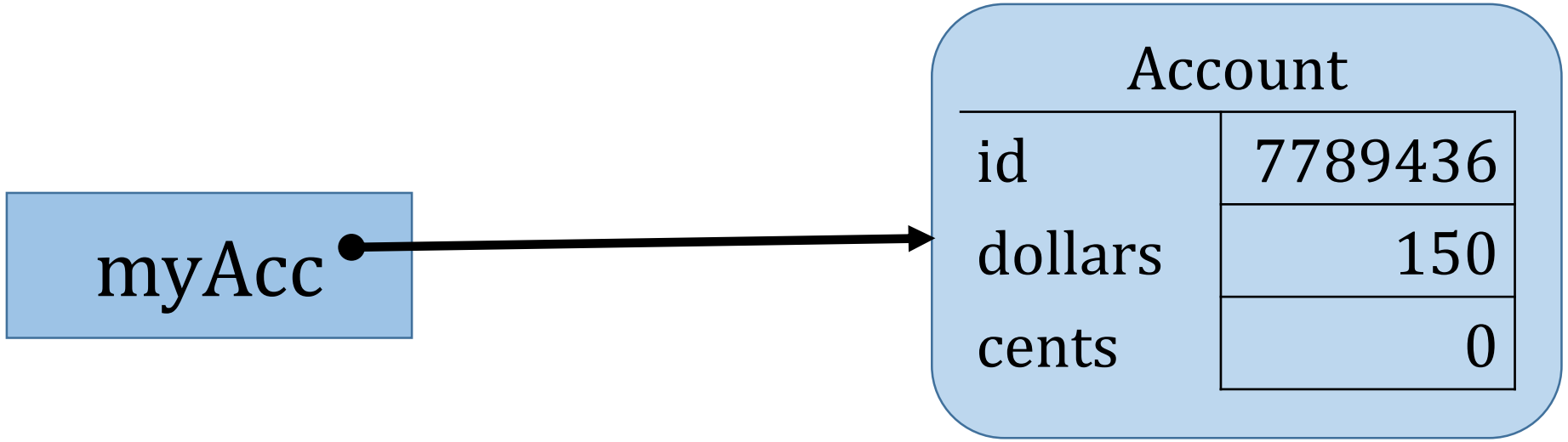
```
Counter myCounter = new Counter();
```



- 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 Variables Schematically

```
Account myAcc = new Account(150.0);
```



Reference Variables

- Handles on the encapsulated object
- 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
- NO ARITHMETIC ON HANDLES!!!!!

Instantiating Reference Variables

```
output = new java.io.PrintWriter(
             new java.io.FileOutputStream("out.txt"));
java.awt.Rectangle rect = new
java.awt.Rectangle(10,30,25,60);
```

- 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

Importing Packages

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

```
import java.io.*;
import java.awt.Rectangle;

output = new PrintWriter( new FileOutputStream("out.txt"));
Rectangle rect = new Rectangle(10,30,25,60);
```

Class (Static) Variables

- If you use the "static" keyword, that changes a field into a "class" variable instead of a normal field
- "Class" variables are shared by all objects
  - Created when the class is loaded – the start of the program
  - Deleted when the class is dismissed – the end of the program
  - May be initialized when declared
- Often used to keep track of things which a higher level class could take care of

Example Class Variable

```
class Widget {
  String wtype;
  String dateMfg;
  static string count=0;
  public Widget(String t) {
    wtype=new String(t);
    count++;
  … }
```

```
  public static void howMany() {
    System.out.println(
    “Built “ + count + “ widgets.”
    );
  }
… }
```