8-21

Data structure

Contain multiple objects - usually of same type

Long[]

ArrayList

int x;

Arraylist has more power than array

Arraylist size can be changed

Describes relationship between objects

You can access each element in ArrayList

A way to process the objects in order

Depth first search

In-order tree traversal

Algorithms + supplemental classes for building and maintaining the structure

class Node {

Animal data;

Node left;

Node right;

}

Conventions for easy communications with other people

-Nodes: hold data, represent relationships

Iterators: present data “in order”

Inheritance

class RedGiant {

private double mass;

```
void setMass(double mass) {
this.mass = mass;
} //setter or mutator
double getMass() {
return this.mass;
} //getter or accessor
}
```

If you have other identical classes with different names

Have a superclass

```
class Star {...class data here…}
```

And then have subclasses

```
class RedGiant extends Star {
}
class WhiteDwarf extends Star {
}
```

The onion diagram

```
class Animal {
String name;
float weight;
}
class Duck extends Animal {
Int nQuacksPerDay;
}
class CartoonDuck extends Duck {
String cartoonist;
}
```

Duck d = new Duck();

Duck daffy = new CartoonDuck()

Java builds onions from the inside out

Object - the heart of the onion

All classes invisibly extend Object

class Robot

Is an abbreviation for

class Robot extends Object

All classes eventually extend object

What it provides

toString()

equals()

hashCode()

What it doesn’t provide

clone()

wait()

notify()

Every constructor of every class begins with a call to a constructor of the superclass

Class WhiteDwarf extends star {

WhiteDwarf() {

super(); ←———------ this is often invisible

}

}

If you omit super(), the compiler invisibly creates a no-args version that calls the supercass’ no-arg ctor

Classes can have multiple ctors

-every ctor must have a different list of arg types (Signatures)

**