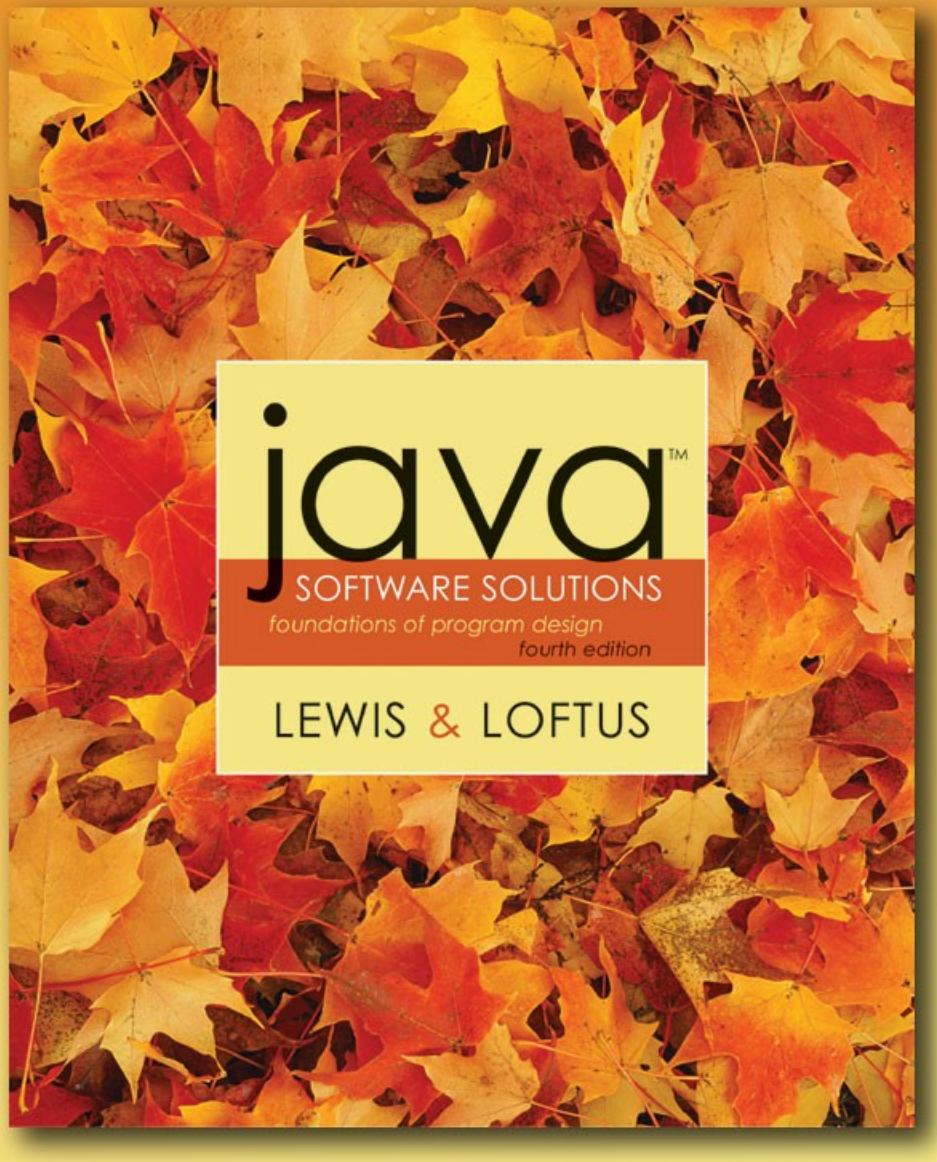
Problem Solving & Object



Following slides are borrowed from the book's Presentation Slides, //duke.csc.villanova.edu/jss1/

PEARSON Addison Wesley

Object Oriented

- ***Abstraction***
- ***Information Hiding - Encapsulation*** : **separate the outside world by its *interface*. Public methods/variables…**
- ***Inheritance*****: a mechanism to allow reuse of the structure and behavior of a class.**
- ***Polymorphism*****: a mechanism to allow different classes respond to same method calls in a similar way with different implementation.**

Creating Objects

- A variable holds either a primitive type or a *reference* to an object
- **Generally, we use the `new` operator to create an object**

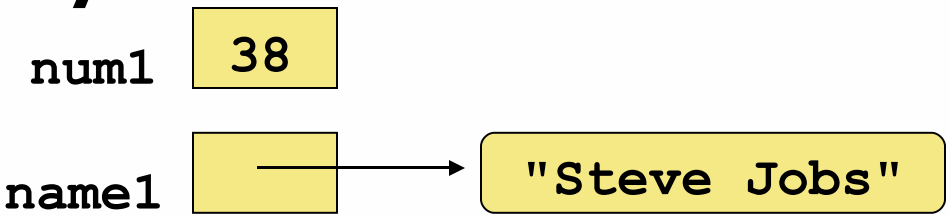
```
title = new String ("Java Software Solutions");
```

This calls the String *constructor*, which is a special method that sets up the object

- Creating an object is called *instantiation.* An object is an *instance* of the class.

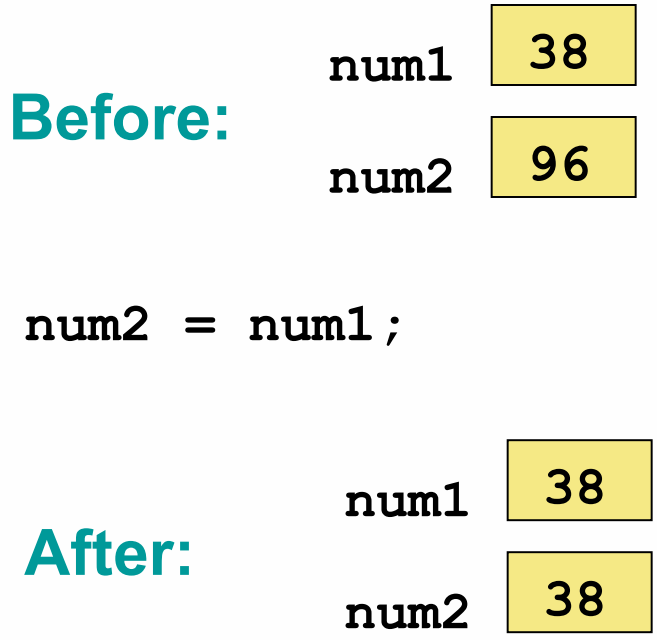
References

- Note that a primitive variable contains the value itself, but an object variable contains the address of the object
- An object reference can be thought of as a pointer to the location of the object
- Rather than dealing with arbitrary addresses, we often depict a reference graphically



Assignment Revisited

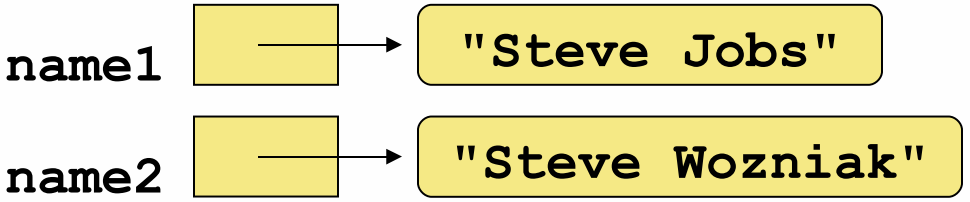
- **The act of assignment takes a copy of a value and stores it in a variable**
- **For primitive types:**



Reference Assignment

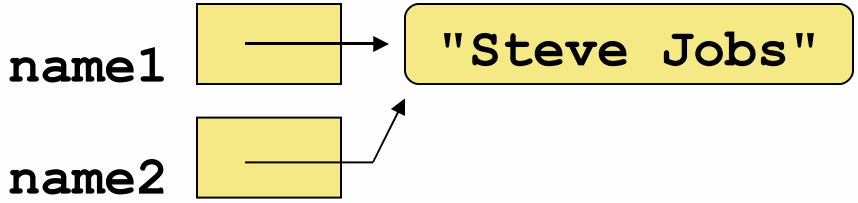
- **For object references, assignment copies the address:**

Before:




```
name2 = name1;
```

After:

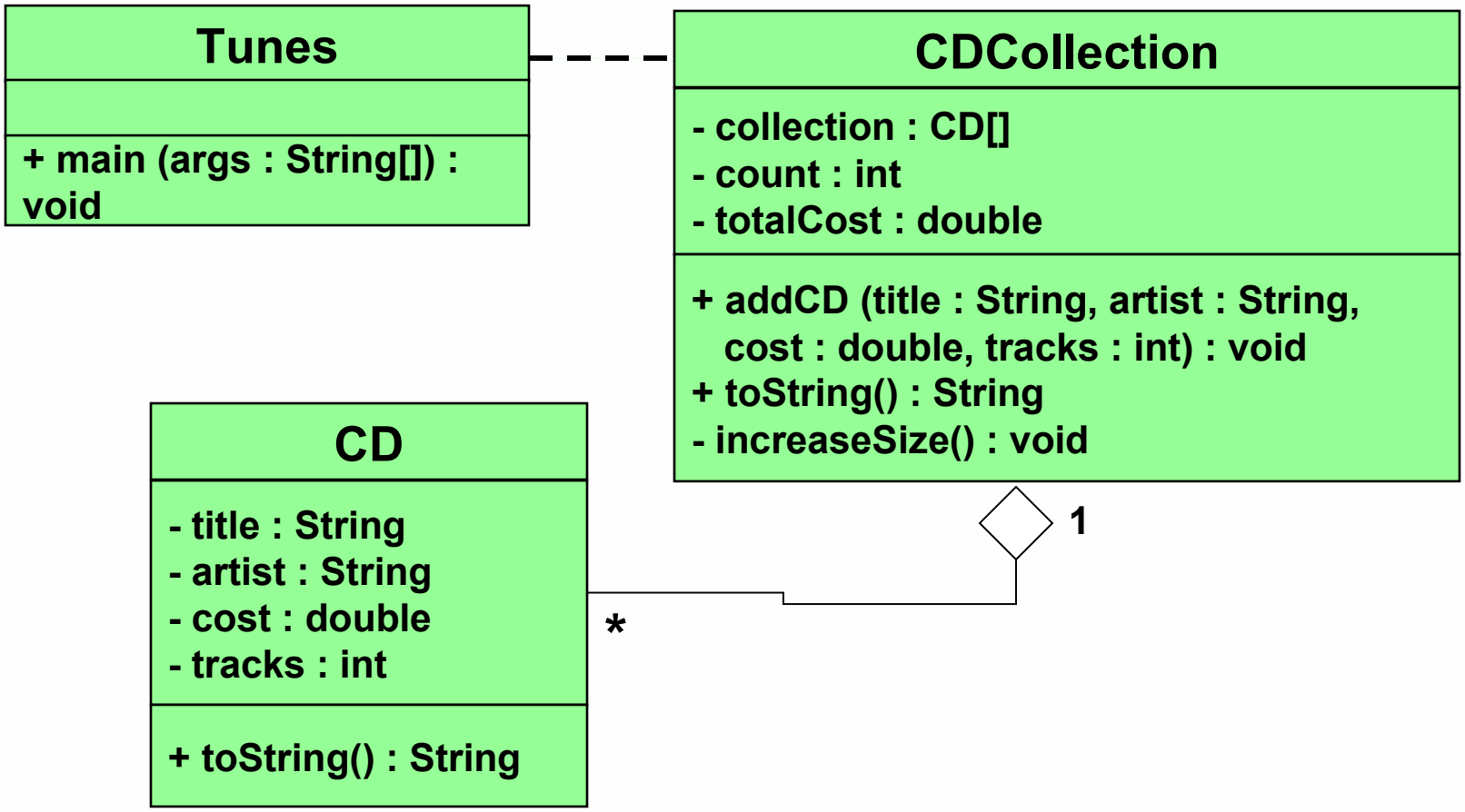


Arrays of Objects

- The following example creates an array of `Grade` objects, each with a string representation and a numeric lower bound
- See `GradeRange.java` (page 384)
- See `Grade.java` (page 385)
- Now let's look at an example that manages a collection of `CD` objects
- See `Tunes.java` (page 387)
- See `CDCollection.java` (page 388)
- See `CD.java` (page 391)

Arrays of Objects

- **A UML diagram for the `Tunes` program:**



Warm up question - 1

```
Open input file
while(still has something to read){                    ...
read one line into a string variable
reverse that string
save it into an ArrayList<String> object 'data'
}
Close the input file;

Open output file
while(ArrayList object 'data' is not empty) {

 get the last element (a string) from 'data'

 write it into output file
}
Close the output file;
```