Review

- Design a class to simulate a bank account
- Implement the class
- Write a demo program that creates bank accounts
- Write junit tests for the bank account class

ISBN 0136130887 © 2008 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

Review

- What data items are associated with a bank account?
  - these are the instance variables
- What kinds of things can you do with a bank account?
  - these are the methods
  - do we need any information to implement the methods besides instance variables?
    - these are parameters to the methods

ISBN 0136130887 © 2008 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

Review

BankAcct

- balance : double

+ getBalance() : double
+ deposit(double amount) : void //positive amount
+ withdraw(double amount) : void //positive amount
+ toString() : String

- Implement the class
- Write a demo program

ISBN 0136130887 © 2008 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

Review

- Demo programs
  - Can be used to test basic functionality
  - Don't test a class thoroughly
- Junit tests
  - Test more thoroughly
  - Make sure the class always behaves as expected

JAVA: An Introduction to Problem Solving & Programming, 5th Ed. By Walter Savitch and Frank Carrano.
ISBN 0136130887 © 2008 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

Review

- Make sure BankAcct.java is selected
- File -> New Junit Test Case
  - Enter BankAcctTest in the box
- Rename testX to testDeposit1
  - Create a BankAcct object called acct
  - Deposit 500
  - Make sure that getBalance returns 500
    - assertEquals(500, acct.getBalance(), 0.001);

*JAVA: An Introduction to Problem Solving & Programming, 5*th Ed. By Walter Savitch and Frank Carrano.
ISBN 0136130887 © 2008 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

Review

- Test method names must start with “test”
- Write a test method called testDeposit2
  - Create a BankAcct object
  - Deposit a negative amount
  - Assert that the balance is still 0:

```
assertEquals(0, acct.getBalance(), 0.001);
```

- Write 2 similar tests for the withdraw method

ISBN 0136130887 © 2008 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

Review

- javadoc comment for a method:
  - Description of what the method does
  - @param tag (one for each parameter)
  - @return tag (if the method is not void)

```
/**
 * Deposit positive amount into this account
 * @param amount the amount to deposit
 */
public void deposit(double amount)
 {
     // implementation
}
```

```
/**
 * Get the balance of this account
 * @return this account's balance
 */
public double getBalance()
 {
     return balance;
}
```


ISBN 0136130887 © 2008 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

Review

- Add a method called `transfer` that transfers a <u>positive</u> amount from this account to another account:

```
/**
 * Transfer positive amount from this account to toAcct
 * @param toAcct the account to transfer to
 * @param amount the amount to transfer
 */
public void transfer(BankAcct toAcct, double amount)
{


}
```


ISBN 0136130887 © 2008 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

Review

- Add a method called `transfer` that transfers a positive amount from this account to another account:

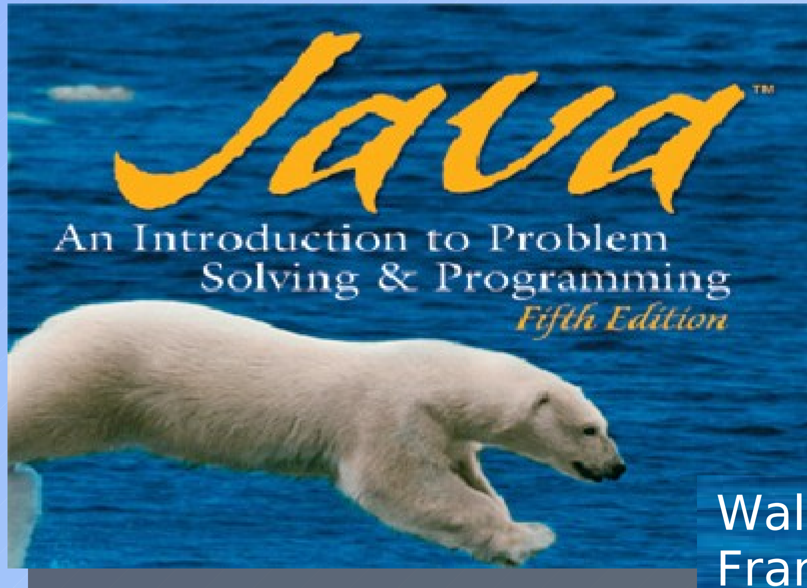
```
public void transfer(BankAcct toAcct, double amount) {
    if (amount >= 0) {
        withdraw(amount); // this.withdraw(amount);
        toAcct.deposit(amount);
    } else {
        System.out.println("Error: unable to transfer " +
            "negative amount.");
    }
}
```


ISBN 0136130887 © 2008 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

Review

- Write tests for the `transfer` method

ISBN 0136130887 © 2008 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved



Walter Savitch
Frank M. Carrano

More About Objects and Methods

Chapter 6

ISBN 0136130887 © 2008 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

Objectives

- Define and use constructors
- Write and use static variables and methods
- Use methods from class **`Math`**
- Use predefined wrapper classes
- Write and use overloaded methods

ISBN 0136130887 © 2008 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

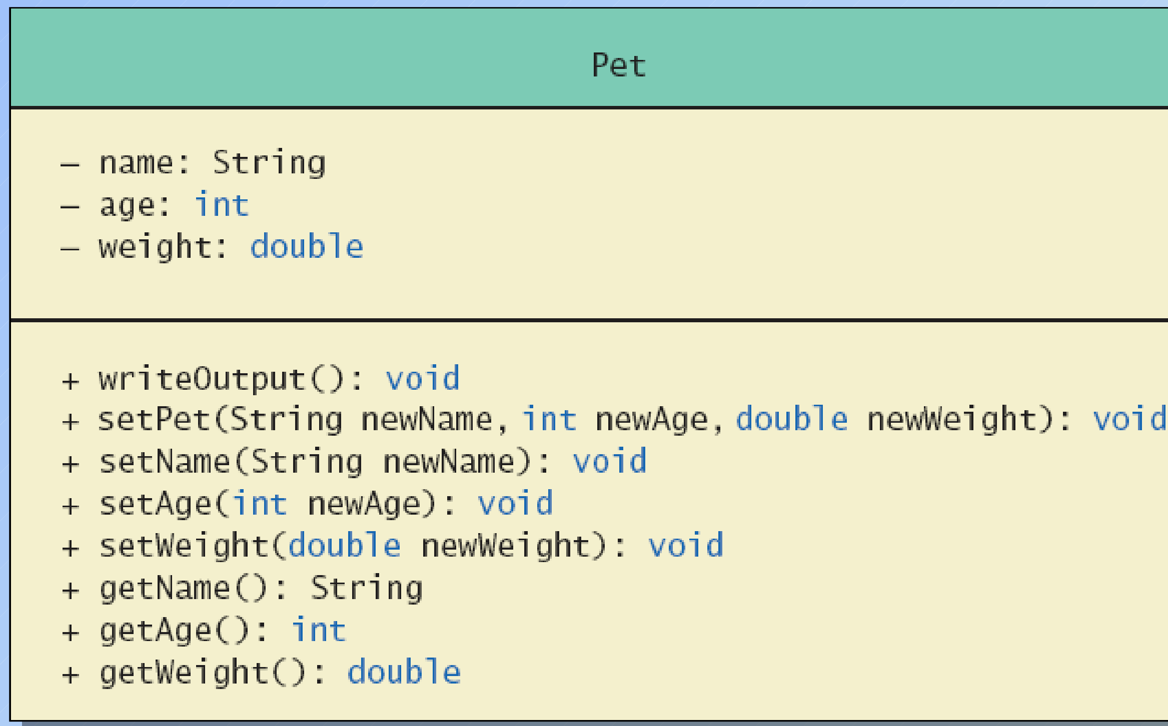
Defining Constructors

- A special method called when instance of an object created with new
  - Creates objects
  - Initializes values of instance variables
- Can have parameters
  - To specify initial values if desired
- May have multiple constructors
  - Each with different numbers or types of parameters

ISBN 0136130887 © 2008 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

Defining Constructors

- Example class to represent pets
- Figure 6.1 Class Diagram for Class **`Pet`**



*JAVA: An Introduction to Problem Solving & Programming, 5*th Ed. By Walter Savitch and Frank Carrano.
ISBN 0136130887 © 2008 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

Defining Constructors

- Download **Pet.java** and **PetDemo.java** from the Examples link on the course webpage
- Note different constructors
  - Default
  - With String parameter
  - With int parameter
  - With double parameter
  - With 3 parameters

ISBN 0136130887 © 2008 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

Defining Constructors

- Constructor without parameters is the default constructor
  - Java will define this automatically if the class designer does not define any constructors
  - If you do define a constructor, Java will not automatically define a default constructor
- Constructors not always included in UML class diagram

ISBN 0136130887 © 2008 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

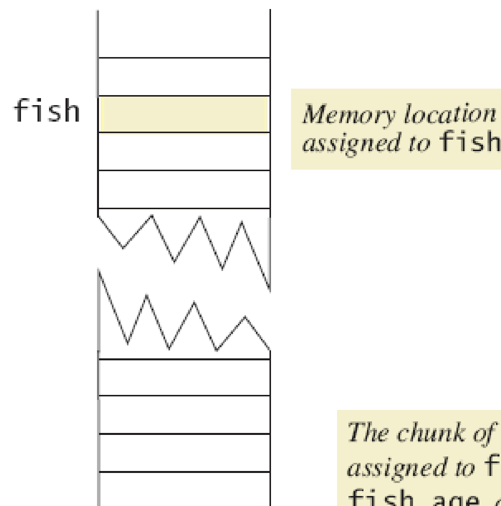
Defining Constructors

- Figure 6.2 A constructor returning a reference

fish = new Pet(“Wanda”, 2, 0.25);

`Pet fish;`

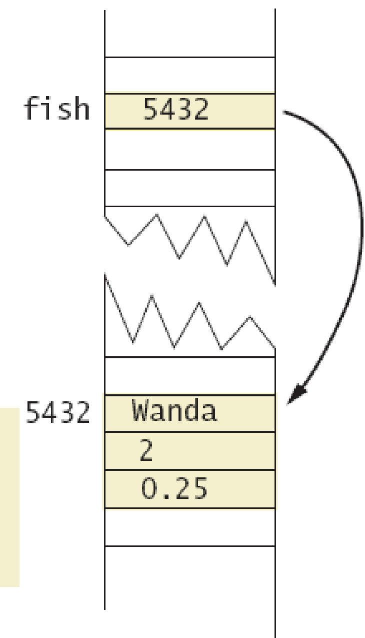
Assigns a memory location to `fish`



~~`fish = new Pet();`~~

Assigns a chunk of memory for an object of the class `Pet`*—that is, memory for a name, an age, and a weight—and places the address of this memory chunk in the memory location assigned to* `fish`

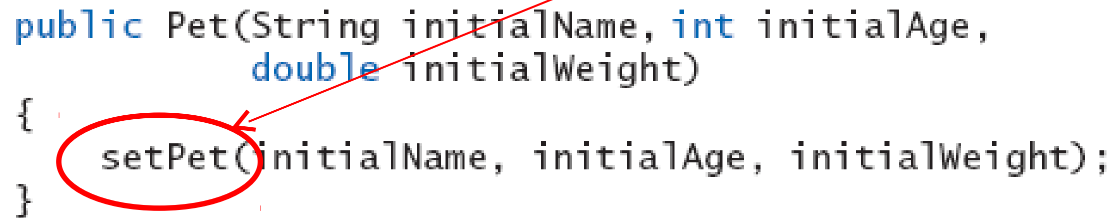
The chunk of memory assigned to `fish.name`*,* `fish.age`*, and* `fish.weight` *might have the address* `5432`.



JAVA: An Introduction to Problem Solving & Programming, 5th Ed. By Walter Savitch and Frank Carrano.
ISBN 0136130887 © 2008 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

Calling Methods from Constructors

- Constructor can call other class methods



```
public Pet(String initialName, int initialAge,
           double initialWeight)
{
    setPet(initialName, initialAge, initialWeight);
}
```

- Change all of the Pet constructors to call the 3-parameter setPet method
  - Error handling is in setAge and setWeight

ISBN 0136130887 © 2008 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

Static Variables & Methods: Outline

- Static Variables
- Static Methods
- Adding a `main` Method to a class
- The `Math` Class
- Wrapper Classes

ISBN 0136130887 © 2008 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

Static Variables

- Static variables are shared by all objects of a class
  - Variables declared `static final` are considered constants – value cannot be changed
- Variables declared `static` (without `final`) can be changed
  - Only one instance of the variable exists
  - It can be accessed by all instances of the class

ISBN 0136130887 © 2008 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

Static Variables

- Static variables also called *class variables*
  - Shared by all instances of the class
  - Contrast with *instance variables* (each object has its own instance variables)
- Both static variables and instance variables are sometimes called *fields* or *data members*

ISBN 0136130887 © 2008 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

Modify BankAcct

- Add 2 constructors to your BankAcct class
  - default constructor (sets balance to 0)
  - a constructor that takes the initial balance as a parameter and sets balance to that

ISBN 0136130887 © 2008 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

Modify BankAcct

- Add an instance variable int acctNum to your BankAcct class
  - Include a getAcctNum method
  - Update your toString method

ISBN 0136130887 © 2008 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

Modify BankAcct

- We want each BankAcct to get a <u>unique</u> account number
- We can do this with a static variable
- Add a static variable nextAcctNum to your BankAcct class:

```
private static int nextAcctNum = 0;
```


ISBN 0136130887 © 2008 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

Modify BankAcct

- Add code in each constructor so that each time an account is created:
  - the account number is set to the next account number
  - the next account number is incremented
- Create at least 3 accounts in your demo program and print them

ISBN 0136130887 © 2008 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

Static Methods

- Some methods may have no relation to any type of object
- Example
  - Compute max of two integers
  - Convert character from upper- to lower case
- Static method declared in a class
  - Can be invoked without using an object
  - Instead use the class name
    - Example: int absValue = Math.abs(8 - 12);

ISBN 0136130887 © 2008 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

Static Methods

- Static methods are not allowed to access instance variables
- Static methods can only call other static methods in the class definition
- *main* is a static method
- All of the methods in the Math class in the java library are static
- See the Math docs in the java API

ISBN 0136130887 © 2008 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

The `Math` Class

- Provides many standard mathematical methods
  - Automatically provided, no import needed
- Example methods, figure 6.3a

Name	Description	Argument Type	Return Type	Example	Value Returned
`pow`	Power	`double`	`double`	`Math.pow(2.0,3.0)`	8.0

`abs`	Absolute value	`int`, `long`, `float`, or `double`	Same as the type of the argument	`Math.abs(-7)`
				`Math.abs(7)`
				`Math.abs(-3.5)`
					7
					7
					3.5

`max`	Maximum	`int`, `long`, `float`, or `double`	Same as the type of the arguments	`Math.max(5, 6)` `Math.max(5.5, 5.3)`	6 5.5

*JAVA: An Introduction to Problem Solving & Programming, 5*th Ed. By Walter Savitch and Frank Carrano.
ISBN 0136130887 © 2008 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

The `Math` Class

- Example methods, figure 6.3b

Name	Description	Argument Type	Return Type	Example	Value Returned
`min`	Minimum	`int`, `long`, `float`, or `double`	Same as the type of the arguments	`Math.min(5, 6)` `Math.min(5.5, 5.3)`	5 5.3
`round`	Rounding	`float` or `double`	`int` or `long`, respectively	`Math.round(6.2)` `Math.round(6.8)`	6 7
`ceil`	Ceiling	`double`	`double`	`Math.ceil(3.2)` `Math.ceil(3.9)`	4.0 4.0
`floor`	Floor	`double`	`double`	`Math.floor(3.2)` `Math.floor(3.9)`	3.0 3.0
`sqrt`	Square root	`double`	`double`	`sqrt(4.0)`	2.0

ISBN 0136130887 © 2008 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

Exercise

- Add a static method to your BankAcct class that transfers a positive amount from one account to another (it's ok to have 2 methods with the same name – more on that soon):

```
public static void transfer(BankAcct from, BankAcct to,
                             double amount)
{

}
```


ISBN 0136130887 © 2008 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

Wrapper Classes

- Recall that arguments of primitive type are treated differently from those of a class type
  - May need to treat primitive value as an object
- Java provides *wrapper classes* for each primitive type
  - Methods provided to act on values

ISBN 0136130887 © 2008 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

Wrapper Classes

- Allow programmer to have an object that corresponds to value of primitive type
- Contain useful predefined constants and methods

ISBN 0136130887 © 2008 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

Wrapper Classes

- Figure 6.4a Static methods in class **`Character`**

Name	Description	Argument Type	Return Type	Examples	Return Value
`toUpperCase`	Convert to uppercase	`char`	`char`	`Character.toUpperCase('a')` `Character.toUpperCase('A')`	`'A'` `'A'`
`toLowerCase`	Convert to lowercase	`char`	`char`	`Character.toLowerCase('a')` `Character.toLowerCase('A')`	`'a'` `'a'`
`isUpperCase`	Test for uppercase	`char`	`boolean`	`Character.isUpperCase('A')` `Character.isUpperCase('a')`	`true` `false`

ISBN 0136130887 © 2008 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

Wrapper Classes

- Figure 6.4b Static methods in class **`Character`**

Name	Description	Argument Type	Return Type	Examples	Return Value
`isLowerCase`	Test for lowercase	`char`	`boolean`	`Character.isLowerCase('A')` `Character.isLowerCase('a')`	`false` `true`
`isLetter`	Test for a letter	`char`	`boolean`	`Character.isLetter('A')` `Character.isLetter('%')`	`true` `false`
`isDigit`	Test for a digit	`char`	`boolean`	`Character.isDigit('5')` `Character.isDigit('A')`	`true` `false`
`isWhitespace`	Test for whitespace	`char`	`boolean`	`Character.isWhitespace(' ')` `Character.isWhitespace('A')`	`true` `false`

Whitespace characters are those that print as white space, such as the blank, the tab character (`'\t'`), and the line-break character (`'\n'`).

ISBN 0136130887 © 2008 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

Overloading: Outline

- Overloading Basics
- Overloading and Automatic Type Conversion
- Overloading and the Return Type

ISBN 0136130887 © 2008 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

Overloading Basics

- When two or more methods have same name within the same class it is called *overloading*
- Java distinguishes the methods by number and types of parameters
  - If it cannot match a call with a definition, it attempts to do type conversions
- A method's name and number and type of parameters is called the *signature*

ISBN 0136130887 © 2008 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

Overloading Basics

- We have been using overloaded methods all along
- In the String class:
  - myString.substring(3);
  - myString.substring(0, 5);
- In the PrintStream class:
  - System.out.println(42);
  - System.out.println("Hello");

ISBN 0136130887 © 2008 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

Overloading Basics

- Download **`class Overload`**
- Note overloaded method **`getAverage`**

```
average1 = 45.0
average2 = 2.0
average3 = b
```

Sample screen output

ISBN 0136130887 © 2008 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

Overloading Basics

- Overloaded constructors or methods must have
  - different number of parameters:

```
public String substring(int startIndex) {
public String substring(int startIndex, int endIndex) {
```

  - OR different types of parameters

```
public void println(int x) {
public void println(double d) {
public void println(boolean b) {
public void println(String s) {
```


ISBN 0136130887 © 2008 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

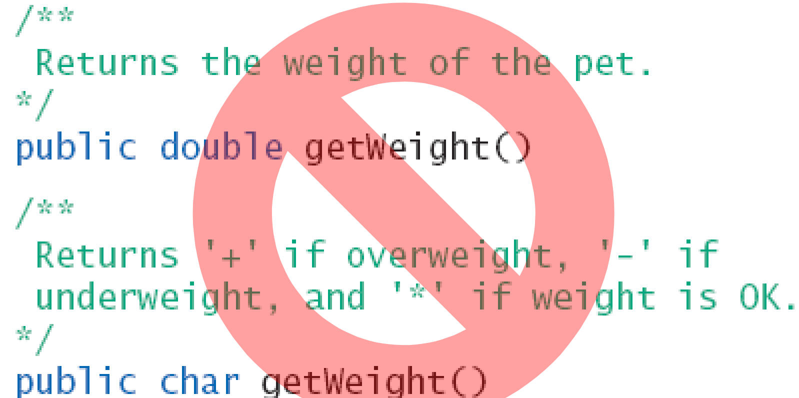
Overloading and Type Conversion

- Overloading and automatic type conversion can conflict
- Recall definition of **`Pet`** class
  - If we pass an integer to the constructor we get the constructor for age, even if we intended the constructor for weight
  - Would be better not to include contructors that take age and weight only
- Remember the compiler only does type conversion if an exact match is not found

ISBN 0136130887 © 2008 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved

Overloading and Return Type

- You can not overload a method where the only difference is the type of value returned



```
/**
 Returns the weight of the pet.
*/
public double getWeight()

/**
 Returns '+' if overweight, '-' if
 underweight, and '*' if weight is OK.
*/
public char getWeight()
```


ISBN 0136130887 © 2008 Pearson Education, Inc., Upper Saddle River, NJ. All Rights Reserved