CSE 21
Intro to Computing II

Lecture 7 – Object Oriented Programming (3)

Announcement

- Lab 6 due before start of next lab
  - Type your answers in a text file and submit it as an attachment
- Project #1 out this Friday (9/30)
  - Due Friday (10/14) at 11:59PM
- Mid-term Exam on 10/19
  - Review during labs next week
- Reading assignment
  - Chapter 10.1 to 10.5 of textbook

Date Class Definition

```
public class Date {
 public int day;
 public int month;
 public int year;
 public Date() {                                    // Constructor 1
   day = month = year = 0;
 }
 public Date(int year) {                            // 2
  day = month = 0;
  year = year;
 }
 public Date(int year, int month) {                 // 3
  day = 0;
  month = month;
  year = year;
 }
 public Date (int year, int month, int day) {       // 4
  day = day;
  month = month;
  year = year;
 }
}
```

We use "**this**" to explicitly access instance variables.

Date Class Definition

```
public class Date {
 public int day;
 public int month;
 public int year;
 public Date() {                                        // Constructor 1
   day = month = year = 0;
 }
 public Date(int year) {                                // 2
  day = month = 0;
  this.year = year;
 }
 public Date(int year, int month) {                     // 3
  day = 0;
  this.month = month;
  this.year = year;
 }
 public Date (int year, int month, int day) {           // 4
  this.day = day;
  this.month = month;
  this.year = year;
 }
}
```

We use "**this**" to explicitly access instance variables.

What's wrong?

```
johnny = new Date();
johnny.month = 27;
johnny.day = -12;
johnny.year = 99999999;

johnny = new Date (13);
johnny = new Date (13, 13);
johnny = new Date (13, 13, 13);
johnny.year = 100;
```

Defensive Programming

```
public class Date {
   public Date (int month) {
    setMonth(month);
   }

   public void setMonth(int month) {
      if (month > 0 && month <= 12)
         this.month = month;
      else
         System.out.println("Invalid month");
   }
}
```

Incorporate error-checking mechanism!

Using dot to Access Everything

```
Date johnny = new Date( );

// instead of johnny.month = 7;
johnny.setMonth(7);  // method call

// month is a variable
System.out.println("This person was born in
  month #" + johnny.month);
```

Mutator Methods

- This type of function allows us to incorporate error checking with our data members
- We can choose to only modify a data member value if the new value meets certain constraints
- But by itself, a “set” function does not prevent another programmer from placing bad values into the data member by using the dot-notation.
  - To enforce this, we can change the access level of our data members to “private”

Private Access-level

```
public class Date {
   private int month;
   private int day;
   private int year;

   public void setMonth(int month) {
      if (month > 0 && month <= 12)
         this.month = month;
      else
         System.out.println("Invalid month");
   }
}
```

Restricted Access

- The “private” access level means that only code belonging directly to the class may use that data member directly.
- All other code must access that data member through some public member function.
- Which means our ***println*** will no longer compile:

```
System.out.println(“This person was born in
  month #” + johnny.month);
```

Accessor Methods

- Once a data member has been made private, if we wish to use the value from a field we need to use a retrieval function.
- These are also called ***accessor functions***, and are usually prefaced with the word "get"

```
public int getMonth() {
   return month;
}
```

Use of Get and Set

```
Date johnny = new Date();

johnny.setMonth(7);

System.out.println("This person was born " + "in
month #" + johnny.getMonth());
```

Symmetry

```
public int getMonth( ) {
    return month;
}

public void setMonth(int month) {
    if (month > 0 && month <= 12)
        this.month = month;
    else
        System.out.println("Invalid month");
    }
```

get has no params

set has no return value

Symmetry

```
public int getMonth( ) {
    return month;
}

public void setMonth(int month) {
    if (month > 0 && month <= 12)
        this.month = month;
    else
        System.out.println("Invalid month");
    }
```

get returns a value of
the data member's type

set takes in a value of
the data member's type

Naming Convention

- The use of words “set” and “get” are not required (*setLastName, getPreviousValue*)
  - A standard style many programmers follow
- Could call them “foo” and “bar”
  - but that would not reflect what the functions are meant to accomplish

Other member methods

- Class methods can do anything we want

```
public class Date {
   private int month;
   private int day;
   private int year;

   public void display( ) {
      System.out.print(month + "/");
      System.out.print(day + "/");
      System.out.println(year);
   }
}
```

What is missing here?

Take a closer look

```
public void display( ) {
   System.out.print(this.month + "/");
   System.out.print(this.day + "/");
   System.out.println(this.year);
}
```

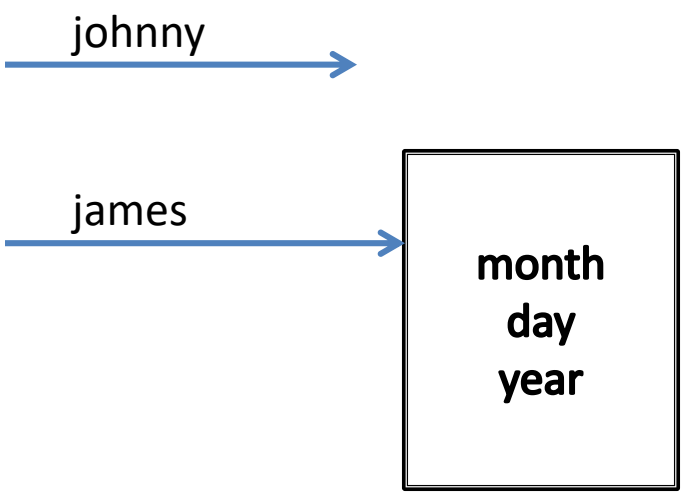
From whence the data?

- We never specify whose day, month, and year to use!
- Actually, we do
  - because you cannot call a class function without a class object to use it on:
    - `johnny.display();`
- Each method call knows which variable to use
  - If we want to work with the data in the Johnny variable, we specify that:
    - `johnny.display();`
  - If we want to work with the data in the Zachary variable, we specify that also:
    - `zachary.display();`

We may not need “this” in the body of the method.

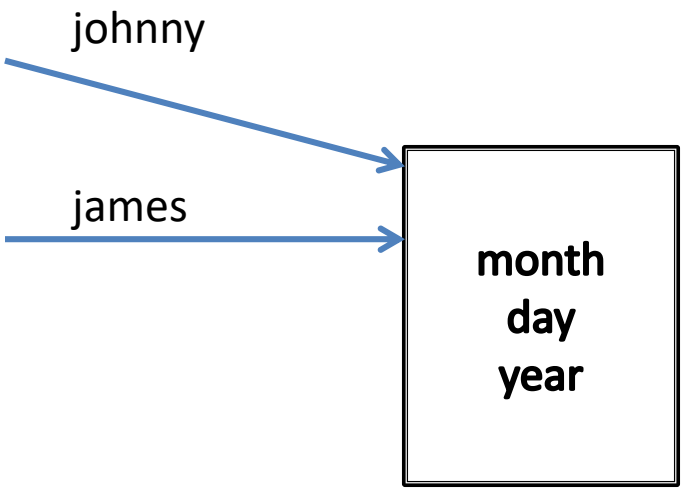
Objects and Pointers

- Date johnny;
- Date james = new Date();
- johnny = james;



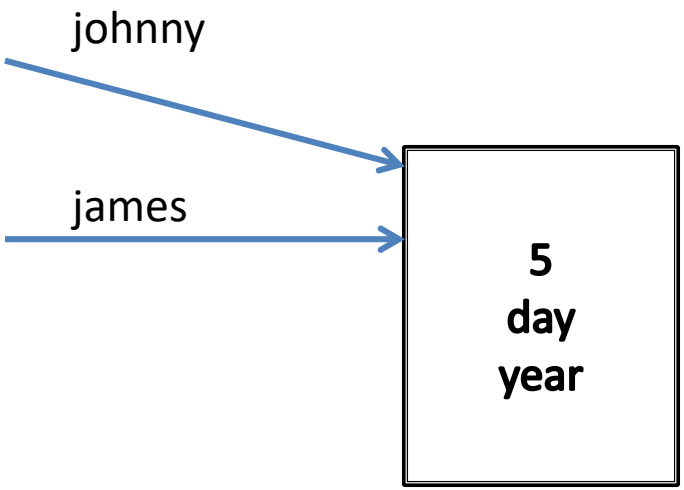
Objects and Pointers

- Date johnny;
- Date james = new Date();
- johnny = james;



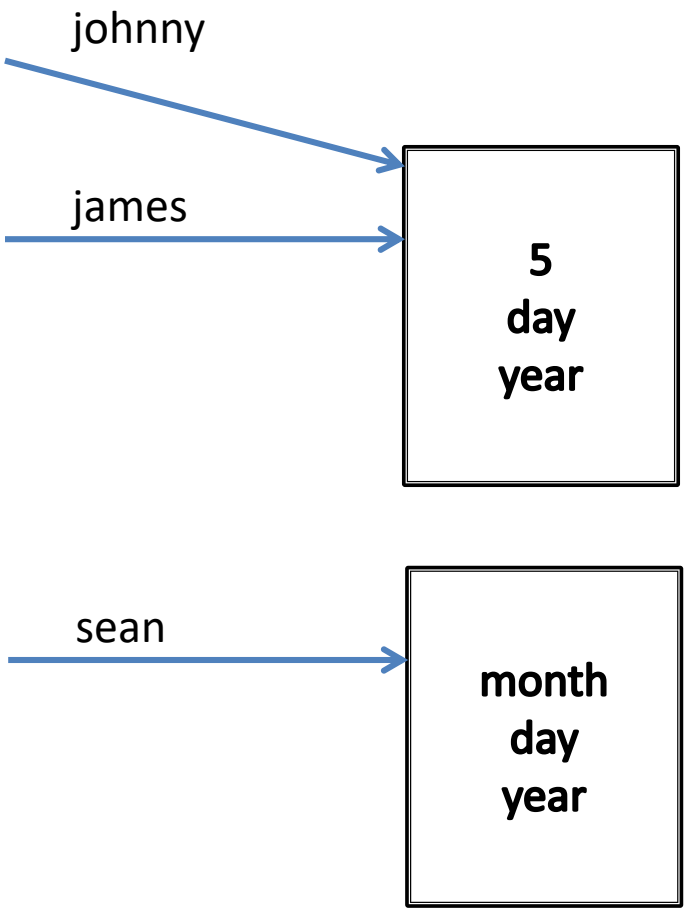
Objects and Pointers

- Date johnny;
- Date james = new Date();
- johnny = james;
- james.setMonth(5);
- johnny.getMonth(); // ?? 5



Objects and Pointers

- Date johnny;
- Date james = new Date();
- johnny = james;
- james.setMonth(5);
- Date sean = new Date();
- sean = james;



Objects and Pointers

- Date johnny;
- Date james = new Date();
- johnny = james;
- james.setMonth(5);
- Date sean = new Date();
- sean = james;