



# More on Objects




February 2, 2012

## **CMPSCI 121, Spring 2012**

*Introduction to Problem Solving with Computers*

Prof. Learned-Miller

# Logistics

Assignment	Due Date
<a href="#">Name and Height</a>	2/2/2012 11:30 PM
 <a href="#">eBook - Chapter 2: Objects and Classes</a>	2/2/2012 11:30 PM
 <a href="#">Chapter 1 Exercises</a>	2/3/2012 11:30 PM
 <a href="#">Chapter 2 Exercises</a>	2/3/2012 11:30 PM
<a href="#">eBook - Chapter 3: Classes, Strings, and I.O.</a>	2/7/2012 11:30 PM
<a href="#">Chapter 3 Exercises</a>	2/8/2012 11:30 PM
<a href="#">Bank Accounts</a>	2/9/2012 11:30 PM

# Logistics

Assignment		Due Date
Name and Height	<b>TODAY!</b>	2/2/2012 11:30 PM
eBook - Chapter 2: Objects and Classes	<b>TODAY!</b>	2/2/2012 11:30 PM
Chapter 1 Exercises	<b>tomorrow</b>	2/3/2012 11:30 PM
Chapter 2 Exercises	<b>tomorrow</b>	2/3/2012 11:30 PM
eBook - Chapter 3: Classes, Strings, and I.O.	<b>Next week</b>	2/7/2012 11:30 PM
Chapter 3 Exercises		2/8/2012 11:30 PM
Bank Accounts		2/9/2012 11:30 PM

# Assignment tip

- `int x;`
- `x = 20;`
- `x = x + 1;`
- `x = x + x + x;`

After all of this, what is x?

# Errors

- Programming is fraught with errors.
- It is important to understand some of the different kinds of errors.

# Errors, in English.

- “Abraham Lincoln was born in 1994.”
  - A factual error.
- “abraham lincoln was born in 1809.”
  - Grammatical error.
- “Babraham Lincoln was never born.”
  - No error.

# Errors, in Java

■ `double cm = mm x 10;`

■ Grammatical error (“syntax error”).

An error in the structure of the statement.

■ `double cm = mm * 7.5;`

■ Logical error: an error in reasoning or logic.

■ `double cm = mm * .1;`

■ no error

# Syntax (grammar) in Java

- Semicolons at the end of each statement.
- Strings must be enclosed in “”.
- Every ( must be matched by a ).
  - parentheses
- Every { must be matched by a }.
  - curly braces
- and many other....



# Declarations

- `int x;`
- `String name;`
- `Infant emmaLM;`

# Assignments

- `x = 3;`
- `name = "Erik";`
- `emmaLM = new Infant("Emma",2);`

# Combining declarations and assignments

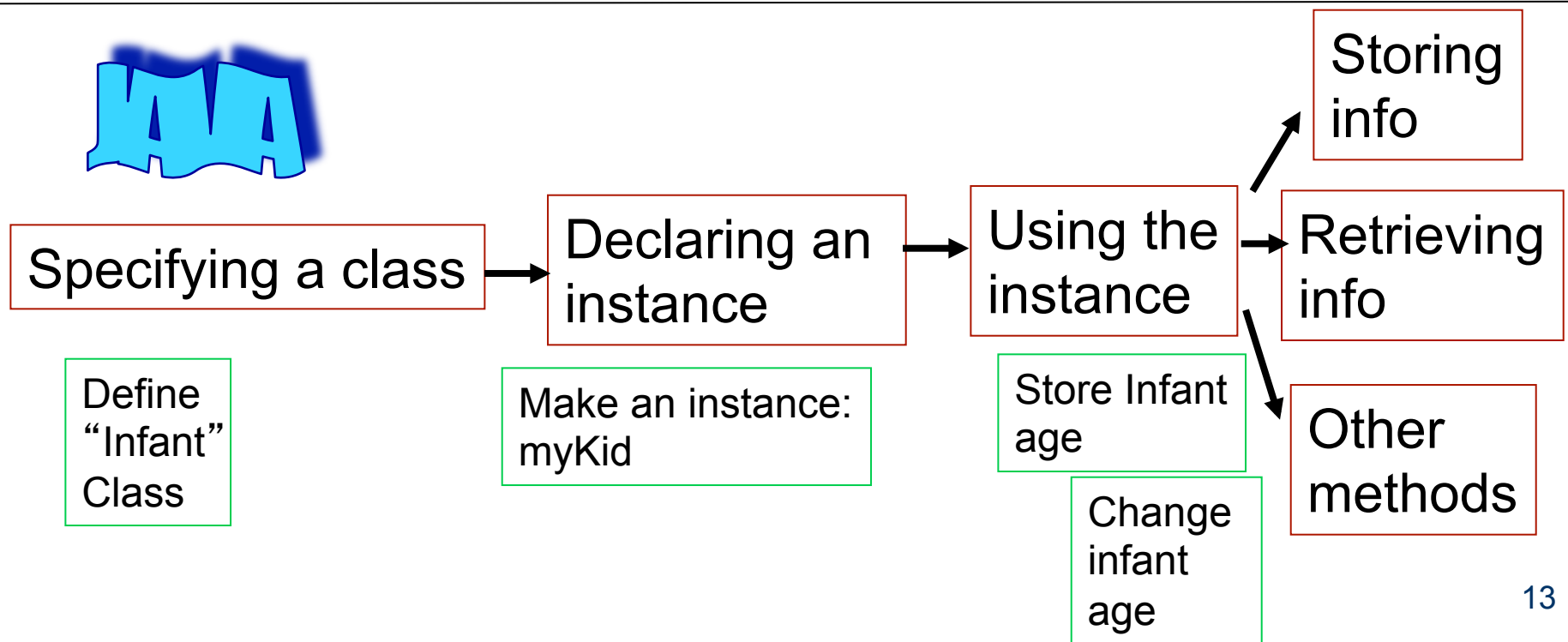
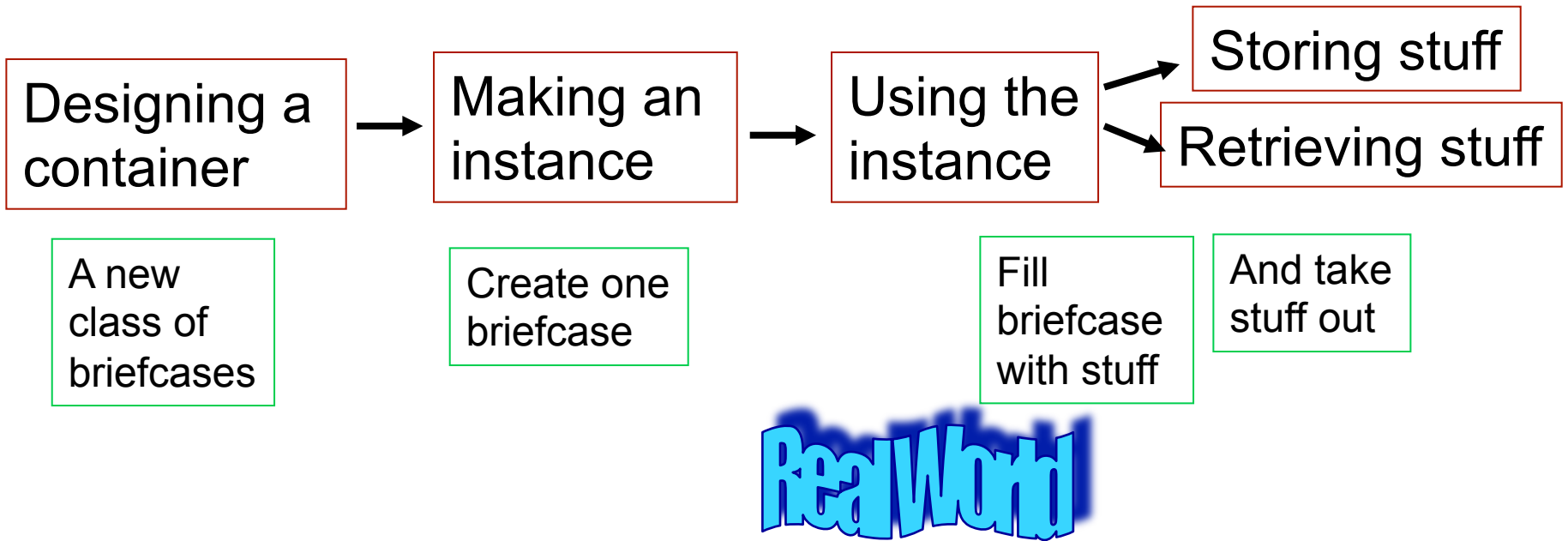
- `int x = 3;`
- `String name = "Erik";`
- `Infant emmaLM = new Infant("Emma",2);`

# Back to objects

- Creating objects with `new` command
- Attributes
- Methods
  - Calling methods with values
  - Calling methods without values

```
1 public class InfantTester{
2
3     public static void main (String[] args){
4         Infant myKid = new Infant("Lizzie",4);
5         int lizAge = myKid.getAge();
6         System.out.println("my kid's name is " + myKid.getName());
7         myKid.anotherMonth();
8         System.out.println("my kid is now " + myKid.getAge() + " months");
9     }
10 }
```

```
1 public class Infant{
2
3     private String name;
4     private int age; // age in months
5
6     public Infant(String who, int months){
7         name = who;
8         age = months;
9     }
10
11     public String getName(){return name;}
12
13     public int getAge(){return age;}
14
15     public void anotherMonth(){age = age + 1;}
16 }
```



# Creating objects with **new**

```
new Infant("Lizzie", 4);
```

# Creating objects with **new**

```
new Infant("Lizzie", 4);
```



The class of the newly created object.



# Creating objects with **new**

```
new Infant("Lizzie", 4);
```



Information needed by the “constructor”.  
Called “arguments” or “parameters”.

What name should I stitch on the briefcase?  
How many pockets should the backpack have?

# Creating objects with **new**

```
new Infant("Lizzie", 4);
```

Every *class* of objects (like `Infant`), needs certain information when it creates a new *instance*.

The number of parameters or arguments may be:

2

7

312

0

# Creating objects with `new`

```
new Infant("Lizzie", 4);
```

The meaning of the arguments or parameters depends upon the *class* definition.

## Creating objects with **new**

```
new Infant("Lizzie", 4);  
new Infant("baby", 7);  
new Infant(3, "bobbie"); // error  
new Infant("Lizzie"); // error
```

# Creating objects with `new` and assigning them to a variable.

```
lizzie = new Infant("Lizzie", 4);  
baby = new Infant("baby", 7);  
bob = new Infant(3, "bob"); // error  
liz = new Infant("Liz"); // error
```

# Creating objects with **new** plus, declaration and assignment!

```
Infant lizzie = new Infant("Lizzie",4);  
Infant baby = new Infant("baby", 7);  
Infant bobbie = new Infant(3,"bobbie"); // error  
Infant lizzie = new Infant("Lizzie"); // error
```

# Creating objects with **new**

```
Car hotrod = new Car("Chevy", 10, 5);
```

# Creating objects with **new**

```
Car hotrod = new Car("Chevy", 10.0, 5.0);
```

This *class* of objects, which is called Car, needs 3 pieces of information to be created, a String, and two decimal numbers.

In this case, they represent the manufacturer, the fuel capacity, and the amount of fuel in the car.



# Constructors

- A piece of code that creates a new instance of a class is called a *constructor*.

# Some Code

```
1 public class InfantTester{
2
3     public static void main (String[] args){
4         Infant myKid = new Infant("Lizzie",4);
5         int lizAge = myKid.getAge();
6         System.out.println("my kid's name is " + myKid.getName());
7         myKid.anotherMonth();
8         System.out.println("my kid is now " + myKid.getAge() + " months");
9     }
10 }
```

A Declaration

Performing an Assignment

Calling a Constructor

# Attributes

- Data stored in your object:
  - Name
  - Age
  - make of car
  - Picture of a person
  - Other objects
  - *Anything at all you want to store about an object!!!*

# Attributes

- File analogy:
  - Anything you would put in a person's file folder.
- Tax form analogy
  - Any piece of information you would enter on a tax form.

```
1 public class InfantTester{
2
3     public static void main (String[] args){
4         Infant myKid = new Infant("Lizzie",4);
5         int lizAge = myKid.getAge();
6         System.out.println("my kid's name is " + myKid.getName());
7         myKid.anotherMonth();
8         System.out.println("my kid is now " + myKid.getAge() + " months");
9     }
10 }
```

```
1 public class Infant{
2
3     private String name;
4     private int age; // age in months
5
6     public Infant(String who, int months){
7         name = who;
8         age = months;
9     }
10
11     public String getName(){return name;}
12
13     public int getAge(){return age;}
14
15     public void anotherMonth(){age = age + 1;}
16 }
```



## Car.java

```
public class Car{
    // the Car attributes
    String make; // manufacturer
    double fuelCapacity;
    double fuelAmount;

    // the Car constructor
    public Car(String what, double cap, double amt){
        make = what;
        fuelCapacity = cap;
        fuelAmount = amt;
    }
    // the Car methods
    public String getMake(){
        return make;
    }
    public double getCapacity(){
        return fuelCapacity;
    }
    public double getFuel(){
        return fuelAmount;
    }
    public void setFuel(double amt){
        fuelAmount = amt;
    }
    public double unusedCap(){
        return (fuelCapacity - fuelAmount);
    }
}
```

```
public class Banking{
    private String name;
    private int checkMoney;
    private int saveMoney;

    public Banking(String who, int checking, int saving){
        name = who;
        checkMoney = checking;
        saveMoney = saving;
    }

    public String getName(){
        return name;
    }

    public int getCheckMoney(){
        return checkMoney;
    }

    public int getSaveMoney(){
        return saveMoney;
    }

    public int getTotalMoney(){
        return(checkMoney+saveMoney);
    }

    public void setCheckMoney(int amt){
        checkMoney = amt;
    }

    public void setSaveMoney(int amt){
        saveMoney = amt;
    }

    public String toString(){
        return(name + " checking: " + checkMoney + " savings: " + saveMoney);
    }
}
```

# “inside” and “outside” of the class definition



```
1 public class InfantTester{
2
3     public static void main (String[] args){
4         Infant myKid = new Infant("Lizzie",4);
5         int lizAge = myKid.getAge();
6         System.out.println("my kid's name is " + myKid.getName());
7         myKid.anotherMonth();
8         System.out.println("my kid is now " + myKid.getAge() + " months");
9     }
10 }
```

```
1 public class Infant{
2
3     private String name;
4     private int age; // age in months
5
6     public Infant(String who, int months){
7         name = who;
8         age = months;
9     }
10
11     public String getName(){return name;}
12
13     public int getAge(){return age;}
14
15     public void anotherMonth(){age = age + 1;}
16 }
```

# Accessing Attributes Directly

- Inside the "class file"
  - Class: Infant      Attribute: age
  - age
    - `System.out.println("age is "+age);`
    - `System.out.println("name is "+name);`

# Accessing Attributes Directly

- Outside the "class file"
  - Class: Infant      Attribute: age
  - Instance: myKid
  - myKid.age
    - System.out.println("age is "+myKid.age);
    - System.out.println("name is "+myKid.name);
  - yourKid.age

# methods

- method: a mini "sub-program" that you run with the data in a specific object
- Example from cooking:
  - Stop what you're doing and go make the Hollandaise sauce!

# methods

- method: a mini "sub-program" that you run with the data in a specific object
- Example from programming:
  - Stop what you're doing and go adjust the age of the Infant named myKid
  - `myKid.anotherMonth();`

# methods

- Today:
  - Focus on using methods
  - *Not on writing methods*
  - *Not on understanding the internal behavior of methods*

# arguments to methods

- Go make some bbq sauce
  - How much?
- Go make 1 quart of bbq sauce
  - `make_bbq_sauce(1qt);`

# method

- `myKid.anotherMonth();`
- `myKid.getAge();`
- `myKid.setAge(5);`
- `x = myKid.getAge();`
- `s = ErikTaxForm.getField35();`
- `ErikTaxForm.setFirstName("Erik");`



```
1 public class InfantTester{
2
3     public static void main (String[] args){
4         Infant myKid = new Infant("Lizzie",4);
5         int lizAge = myKid.getAge();
6         System.out.println("my kid's name is " + myKid.getName());
7         myKid.anotherMonth();
8         System.out.println("my kid is now " + myKid.getAge() + " months");
9     }
10 }
```

```
1 public class Infant{
2
3     private String name;
4     private int age; // age in months
5
6     public Infant(String who, int months){
7         name = who;
8         age = months;
9     }
10
11     public String getName(){return name;}
12
13     public int getAge(){return age;}
14
15     public void anotherMonth(){age = age + 1;}
16 }
```

# Methods: stuff the object can do.

```
1 public class Infant{
2
3     private String name;
4     private int age;    // age in months
5
6     public Infant(String who, int months){
7         name = who;
8         age = months;
9     }
10
11     public String getName(){return name;}
12
13     public int getAge(){return age;}
14
15     public void anotherMonth(){age = age + 1;}
16 }
```

# Methods: stuff the object can do.

```
1 public class Infant{
2
3     private String name;
4     private int age;    // age in months
5
6     public Infant(String who, int months){
7         name = who;
8         age = months;
9     }
10
11     public String getName(){return name;}
12
13     public int getAge(){return age;}
14
15     public void anotherMonth(){age = age + 1;}
16 }
```

# Methods: stuff the object can do.

```
1 public class Infant{
2
3     private String name;
4     private int age;    // age in months
5
6     public Infant(String who, int months){
7         name = who;
8         age = months;
9     }
10
11     public String getName(){return name;}
12
13     public int getAge(){return age;}
14
15     public void anotherMonth(){age = age + 1;}
16 }
```

# Calling a method

```
1 public class InfantTester{
2
3     public static void main (String[] args){
4         Infant myKid = new Infant("Lizzie",4);
5         int lizAge = myKid.getAge();
6         System.out.println("my kid's name is " + myKid.getName());
7         myKid.anotherMonth();
8         System.out.println("my kid is now " + myKid.getAge() + " months");
9     }
10 }
```

# Method call within a print.

```
1 public class InfantTester{
2
3     public static void main (String[] args){
4         Infant myKid = new Infant("Lizzie",4);
5         int lizAge = myKid.getAge();
6         System.out.println("my kid's name is " + myKid.getName());
7         myKid.anotherMonth();
8         System.out.println("my kid is now " + myKid.getAge() + " months");
9     }
10 }
```

# Another method

```
1 public class InfantTester{
2
3     public static void main (String[] args){
4         Infant myKid = new Infant("Lizzie",4);
5         int lizAge = myKid.getAge();
6         System.out.println("my kid's name is " + myKid.getName());
7         myKid.anotherMonth();
8         System.out.println("my kid is now " + myKid.getAge() + " months");
9     }
10 }
```

```
1 public class InfantTester{
2
3     public static void main (String[] args){
4         Infant myKid = new Infant("Lizzie",4);
5         int lizAge = myKid.getAge();
6         System.out.println("my kid's name is " + myKid.getName());
7         myKid.anotherMonth();
8         System.out.println("my kid is now " + myKid.getAge() + " months");
9     }
10 }
```



```
public class Banking{
    private String name;
    private int checkMoney;
    private int saveMoney;

    public Banking(String who, int checking, int saving){
        name = who;
        checkMoney = checking;
        saveMoney = saving;
    }

    public String getName(){
        return name;
    }

    public int getCheckMoney(){
        return checkMoney;
    }

    public int getSaveMoney(){
        return saveMoney;
    }

    public int getTotalMoney(){
        return(checkMoney+saveMoney);
    }

    public void setCheckMoney(int amt){
        checkMoney = amt;
    }

    public void setSaveMoney(int amt){
        saveMoney = amt;
    }

    public String toString(){
        return(name + " checking: " + checkMoney + " savings: " + saveMoney);
    }
}
```

# Summary

- new creates an instance of a particular *class* of object
- attribute is a piece of information about an object

End for today