Projeto de Software / Programação 3
Orientação a Objetos: Conceitos Básicos

Baldoino Fonseca/Marcio Ribeiro
baldoino@ic.ufal.br

Procedural Programming

Function-based

- Programming unit: function
- We group functions to form a program

What is the size of an array?

23	12	25	32	33	53	21	2	…	89

```
function doSomething() {
  $array = array(23,12,25,32,33,53,21,2,...,89);
  for ($i = 0; $i < count($array); $i++) {
    ...
  }
}
```

The array does NOT know!

Use this idea for all objects in the world…

The array could “know” its own size!

23	12	25	32	33	53	21	2	…	89

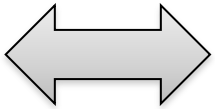
```
count(array);
```

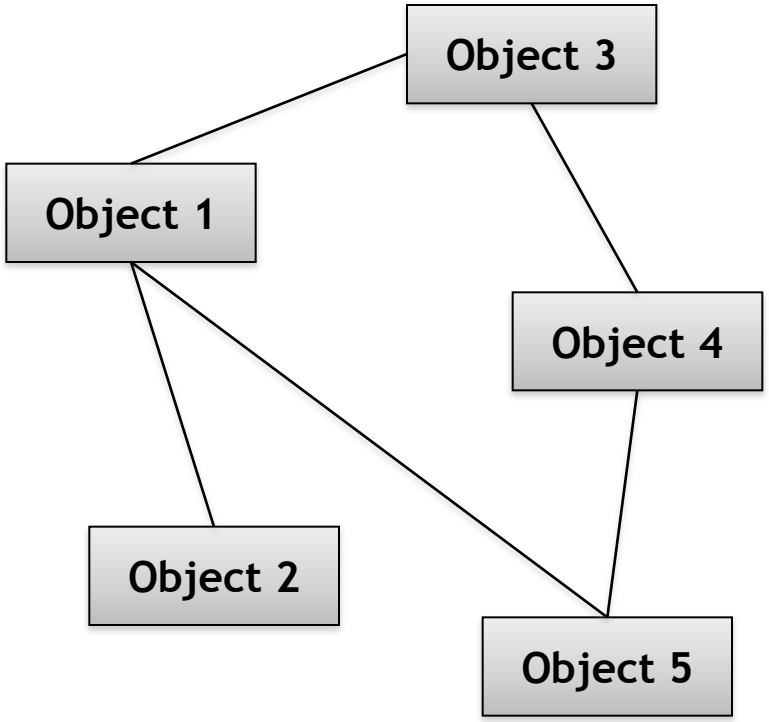
versus

```
array.size();
```

Object-Oriented Programming

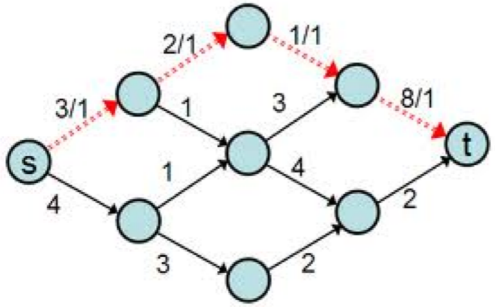
Collection of interacting objects





What can we model with objects?

- **Tangible things**
  - Airplane
  - Seat
- **Conceptual things**
  - Position job (pilot)
  - Date
- **Processes**
  - Sorting passengers
  - Finding a passenger



From *http://www.cs.brown.edu/courses/cs015/lectures.html*

Objects have

- **Properties**
  - Features that describe them
- **Capabilities**
  - What they can do, how they behave

Properties

- Features that describe the object
- Constant or variable
- Properties may themselves be objects

speed

engines

capacity

State

Collection of all of an object's properties and their values; it changes if any property value changes

From *http://www.cs.brown.edu/courses/cs015/lectures.html*

Capabilities

- **Allow objects to perform actions**
  - Initial state
  - Change state
  - Responses based on properties

- **Objects "know" how to do things**

accelerate _____

decelerate _____

_____

Modeling objects

- How to model every object in earth?
  - Commonalities
  - Variabilities

- We need a way to create objects sets taking the advantage of their commonalities

Classification

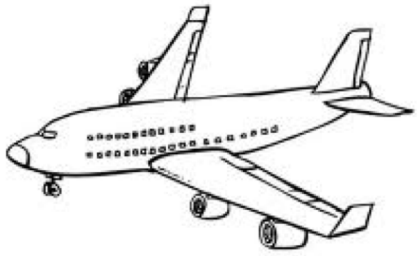
- **Group of objects that has the same set of properties and capabilities**

- **Define an object type: Aircraft**

accelerate
decelerate

Airline company: GOL
Manufacturer: Boeing
Model: 737
N. of engines: 2
Capacity: 174

Airline company: Lufthansa
Manufacturer: Boeing
Model: 747
N. of engines: 4
Capacity: 540



accelerate
decelerate

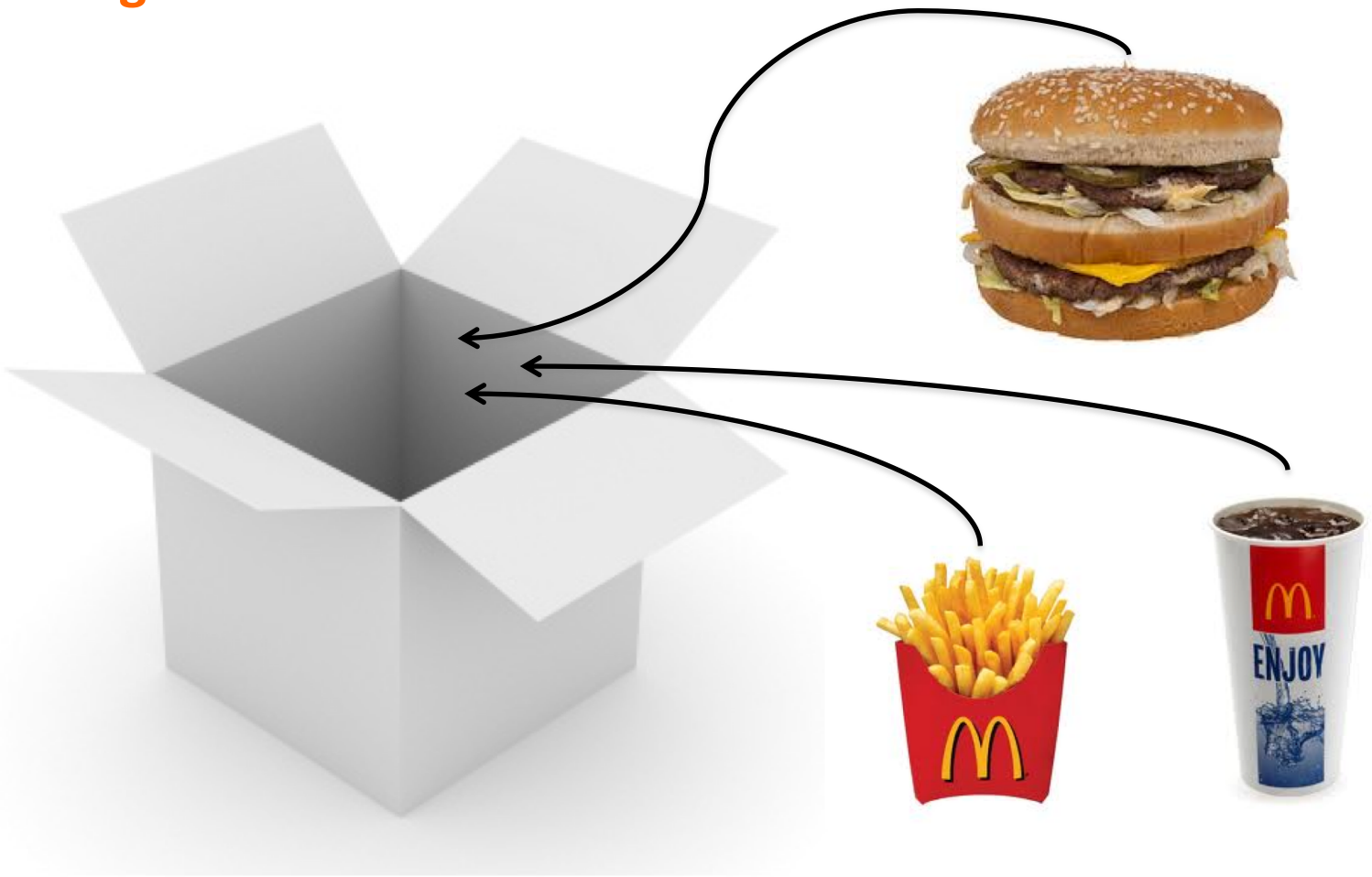
How to create an object?

Think about it

- An object has:
  - Properties
  - Capabilities
- An object know how to do things…
- Properties and capabilities should be “inside” the object
- So, we can use a box to throw everything in it

Let's create a snack object…

- … using a box



We need a TEMPLATE!

Template to organize our snack

Class

- Defines a template for making objects instances
- Defines properties as **attributes**
- Defines capabilities as **methods**

From *http://www.cs.brown.edu/courses/cs015/lectures.html*

Objects instances: individual objects

- Made from class
- Many instances of the same class
- Same attributes, potentially with different values

From *http://www.cs.brown.edu/courses/cs015/lectures.html*

Implementing a Class in Java

Properties (attributes)

Capabilities (methods)

```
public class Aircraft {

    String manufacturer;
    int speed;
    int numberOfEngines;

    public Aircraft(String manufacturer,
                    int speed,
                    int numberOfEngines) {

      this.manufacturer    = manufacturer;
      this.speed           = speed;
      this.numberOfEngines = numberOfEngines;
    }

    public void accelerate(int speed) {
      …
    }

}
```

Notice that…

- There is a special method named “constructor”

- Code conventions!
  - numberOfEngines
  - Aircraft (class); Aircraft (constructor)

How to instantiate an object of Aircraft?!

- **New operator:**
  - Used to create instances
  - We call the constructor capability

```
public class Main {

  public static void main (String[] args) {
    Aircraft boeing737 = new Aircraft("Boeing", 0, 2);

    boeing737.accelerate(840);
  }

}
```

Changing the object state

- **Changing the speed**
  - We do it by calling accelerate…
  - … or by assigning a new value to speed directly

- **Can you see any problem?!**

```
public class Main {

  public static void main (String[] args) {
    Aircraft boeing737 = new Aircraft("Boeing", 0, 2);

    boeing737.accelerate(840);

    boeing737.speed = -840;
  }

}
```

Encapsulation

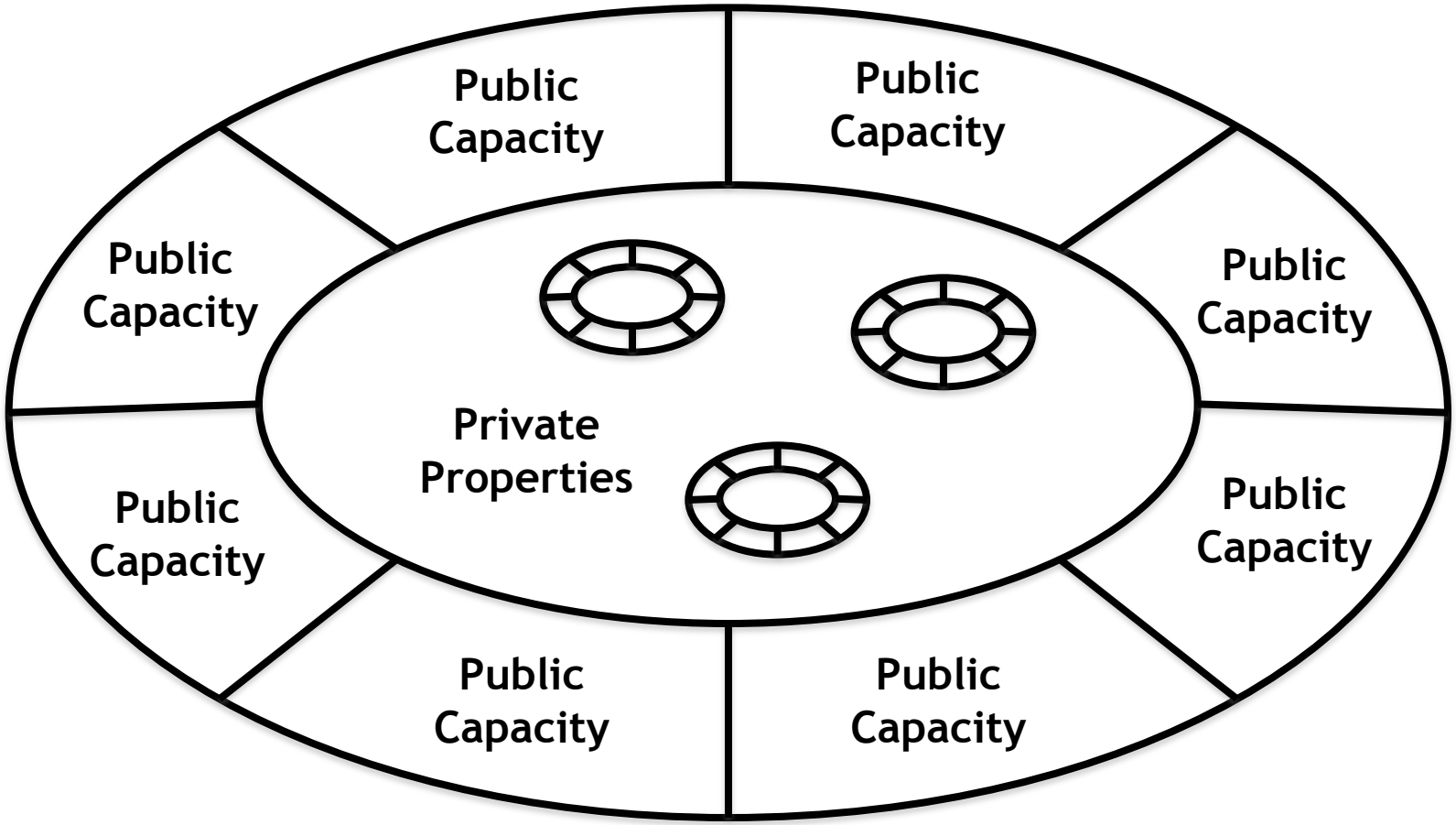
- Access to attributes only through the methods!

- Protects the object state

Protect your attributes!

```
public class Aircraft {

  private String manufacturer;
  private int speed;
  private int numberOfEngines;

  public void accelerate(int speed) {
    if ((speed >= 0) && (speed > this.speed)) {
      this.speed = speed;
    }
  }

}
```

Encapsulation



From *http://www.cs.brown.edu/courses/cs015/lectures.html*

Now…

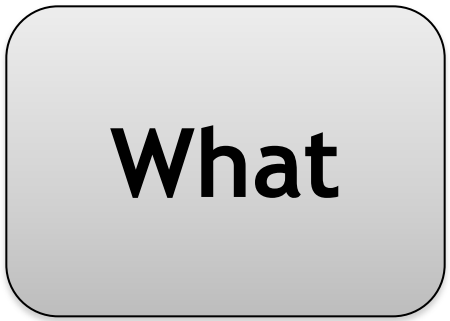
- … negative speeds are not allowed!

```
public class Main {

  public static void main (String[] args) {
    Aircraft boeing737 = new Aircraft("Boeing", 0, 2);

    boeing737.accelerate(-840);

    boeing737.speed = -840;
  }

}
```

Abstraction

- **We don't care about how the methods are implemented!**
  - How does "accelerate" work? WE DON'T CARE!
  - How does "decelerate" work? Again, WE DON'T CARE!

- **So, we hide the strategy used in the implementation**
  - Focus on "what"
  - Forget about "how"



Procedural Programming
versus
Object-Oriented Programming

Array size example

```
function doSomething() {
  $array = array(23,12,25,32,33,53,21,2,...,89);
  for ($i = 0; $i < count($array); $i++) {
    ...
  }
}
```

```
public class Vector {

  private Object[] elementData;

  public int size() {
    …
  }

}
```

Exercises

Exercises

- Think about an object
  - Define its properties
  - Define its capabilities
- Are the properties other objects?
  - If so, define its properties and capabilities as well