Patrick GIRARD
University of Poitiers
2009-2014©

Object-Oriented Programming
in Javatm

Writing Classes and instanciating Objects in Java with BlueJ

Chapter 2 - Section 2

Table of contents

Imperative programming, C

```
#include <stdio.h>
#include <string.h>
struct Person {
  char name[25];
  char company[32]; }
void whoAmI(struct Person p){
  printf("My name is %s\n", p.name);
  printf("I work at %s\n », p.company);
}
void main() {
  struct Person aPerson;
  strcpy(aPerson.name,"JONES");
  strcpy(aPerson.company,"SUN");
  whoAmI(aPerson);
}
```

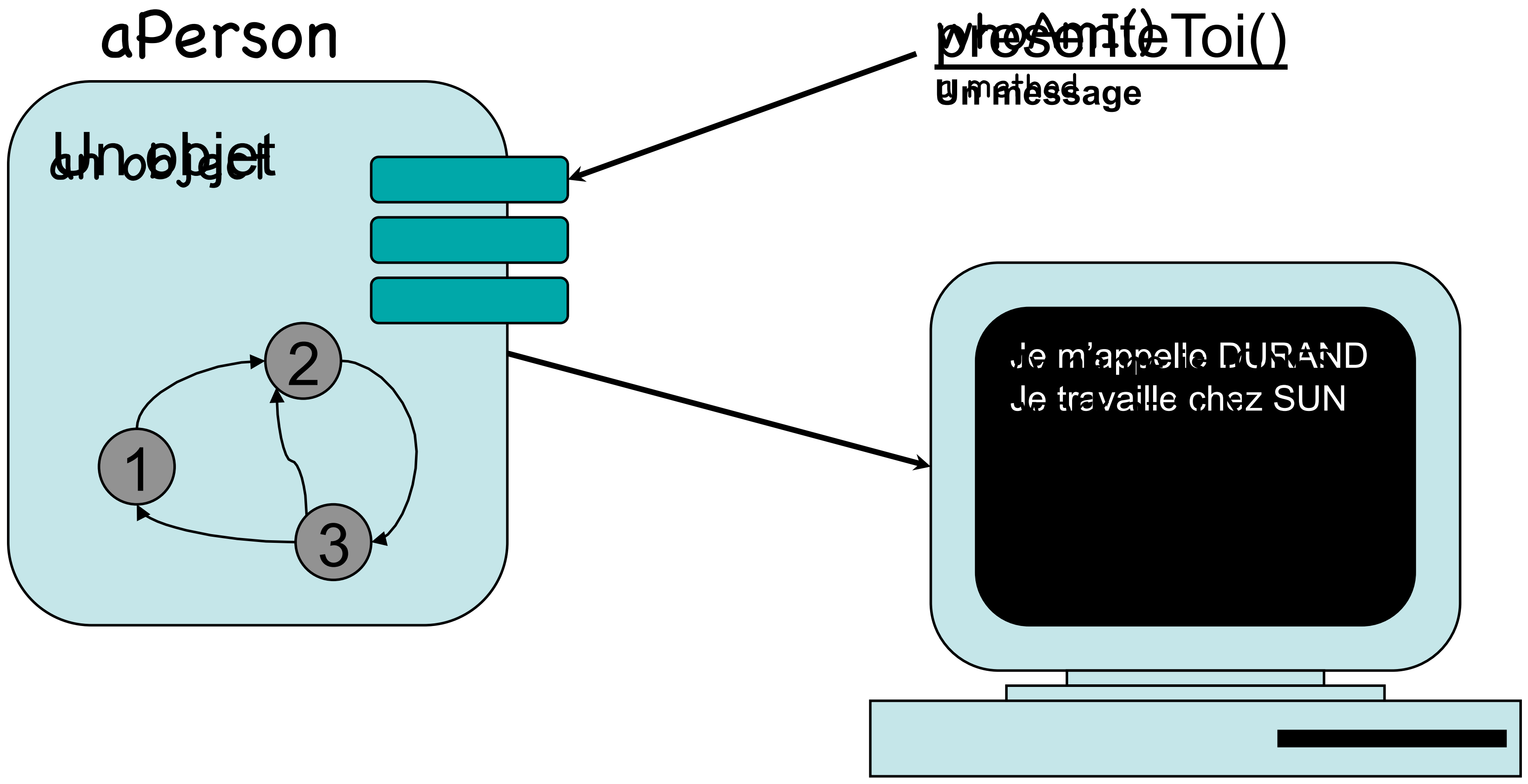
the result

```
Je m'appelle DURAND
Je travaille chez SUN
```

Imperative programming

- Separating Data / Processing
  - struct {} ≠ void whoAmI () {}
- Manipulating objects through functions
  - whoAmI(aPerson)
- Requires knowledge of object structure
  - aPerson.name

Objects


aPerson
Un objet
an object
presenteToi()
whoAmI()
Un message
a method
1
2
3
Je m'appelle DURAND
Je travaille chez SUN

a Java program

```
class Person {
  public String name;
  public String company;
  public void whoAmI(){
    System.out.println("My name is " + name);
    System.out.println("I work at " + company);
  }

class Test {
public static void main(String args[]) {
  Person aPerson;
  aPerson = new Person();
  aPerson.name = "JONES";
  aPerson.company = "SUN";
  aPerson.whoAmI();
}
```

Object-Oriented programming

- Grouping Attributes / Methods
  - class … {
    Attribute
    Method
    }
- Manipulating objects directly
  - aPerson.whoAmI()
- Requires knowledge of public members
  - aPerson.name & aPerson.whoAmI()

Table of contents

A class

name
company

whoAmI()

- Instance attributes
  - public String company;

Attribute's type

Attribute's name

```
class Person {
  public String name;
  public String company;
  public void whoAmI(){
    System.out.println("My name is " + name);
    System.out.println("I work at " + company);
  }
```

A class

name
company

whoAmI()

- Instance attributes
  - Each object instanciated by this class has a value for each attribute
  - public attributes are fully modifiable

name
company

whoAmI()

A class

method name

mandatory

- Methods
  - public void whoAmI() { … }

body

no return value

```
class Person {
  public String name;
  public String company;
  public void whoAmI(){
    System.out.println("My name is " + name);
    System.out.println("I work at " + company);
  }
```

name
company

A class

whoAmI()

Methods

- public methods are the messages each object is able to understand
- methods always have parenthesis () and braces {}
- methods may use parameters
  - public void setName (String name)

    Parameter's type

    Parameter's name
- methods may return values
  - public String getName()

    returned type
- void replaces the returned type when not required

Objects

- Object declaration
  - Person aPerson;
- Object instanciation
  - new Person();
- Object usage
  - aPerson.company …

```
...
  Person aPerson;
  aPerson = new Person();
  aPerson.name = "JONES";
  aPerson.company = "SUN";
  aPerson.whoAmI();
}
```

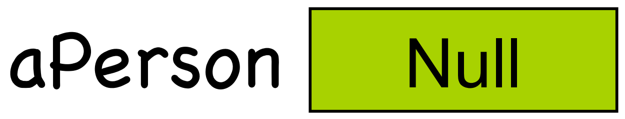
Some tips

- Object programming is dynamic programming but...
  - There is no pointers
  - Object identifiers are called "references"
    - Person aPerson;

References

- Objects are known through their reference
  - null after declaration
    - Person aPerson;
  - Any usage of null reference results in execution error
    - "null pointer exception"

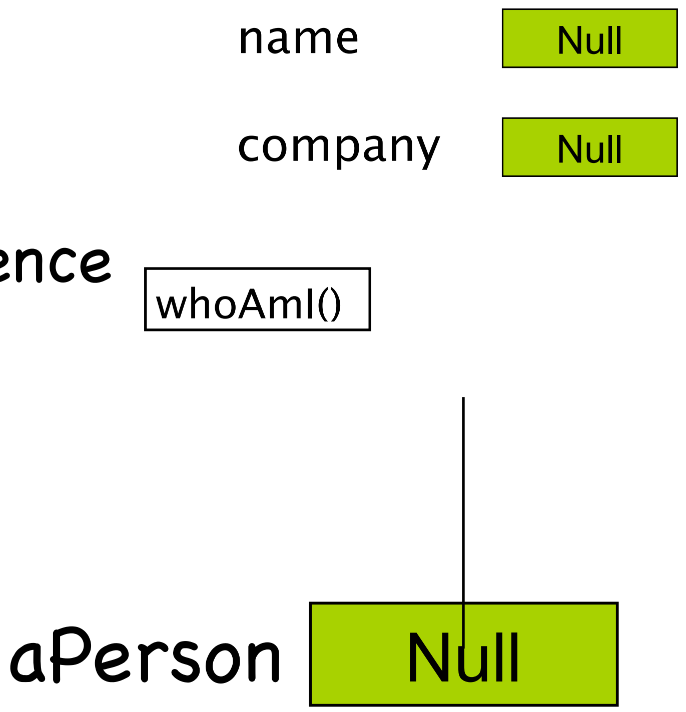
```
Person aPerson;
```



References

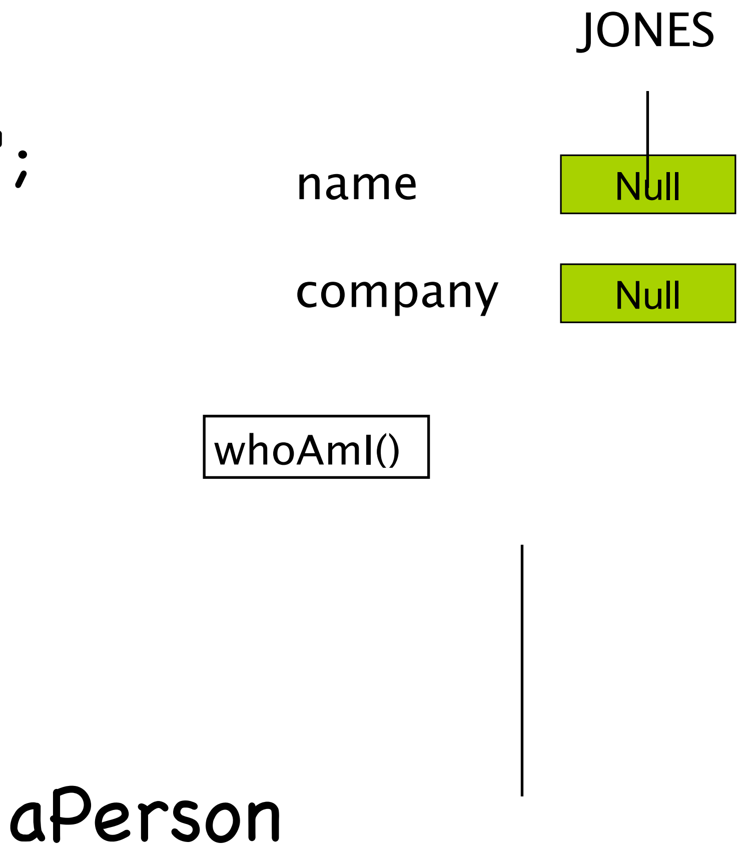
- Instantiation is the actual object creation
  - new creates the objet
    - new Person()
  - Assignment to the reference
    - aPerson = new ...
- Notice...
  - Inside the object...

```
aPerson = new Person();
```



Natural public usage

- Public attributes
  - aPerson.name = "JONES";
- Public methods
  - aPerson.whoAmI();

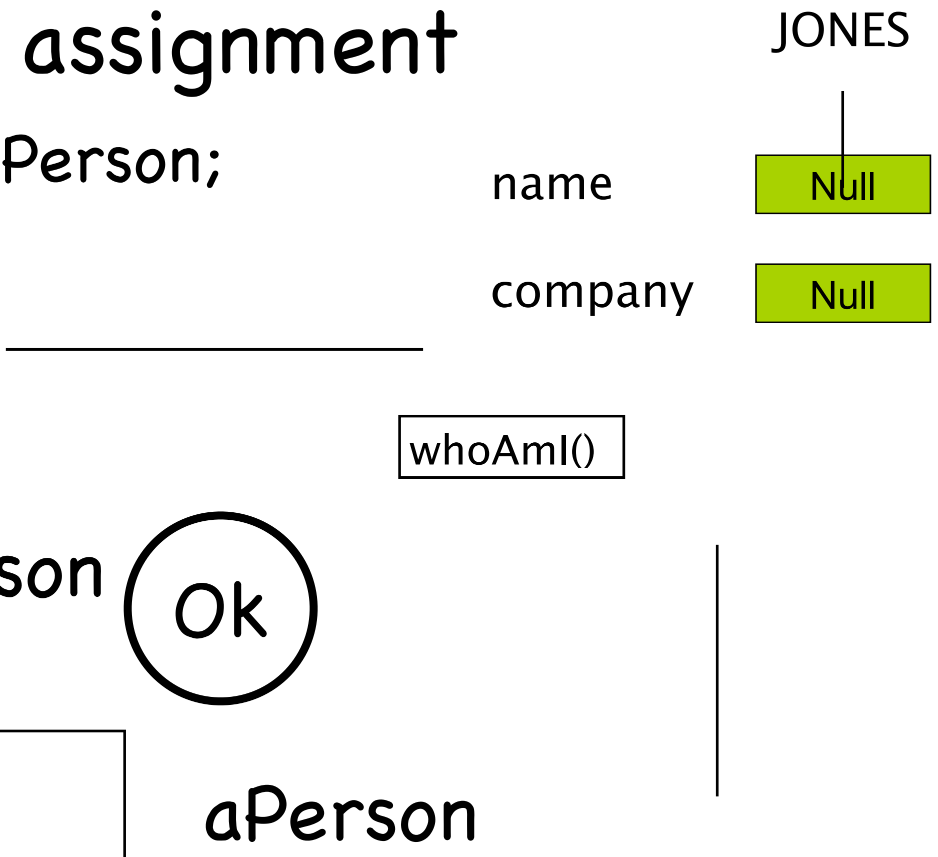


```
aPerson.name = "JONES";
aPerson.whoAmI();
```

Warning

- Be careful with assignment
  - anotherPerson = aPerson;

~~anotherPerson~~



aStudent = aPerson Ok

```
Person anotherPerson
anotherPerson = aPerson;
```

Partial conclusion

- Not so interesting in that way...
  - Objects can be wrong (after declaration)
  - Internal structure must be known
- Why ?
  - Objects do not manage their state !

a keyword: private