Seminar

SeC8

Chapter 8.
Implementation

Software Engineering

Computer Science School

Goals

- Polymorphism – Static and Dynamic binding
- Constructors (invocation order)
- Reuse

- Work (individual / group)

Polymorphism – Static/Dynamic Binding

1. Given the following C# code

```
class Collection{
  private ArrayList icons;
 }
abstract class Icon{
  protected int origin;
  public abstract void draw();
  public int obtainOrigin() {. . .}
}
class RectangularIcon : Icon {
  private int height; private int width;
  public override void draw() {. . .}
  public bool isInside(int p) {. . .}
}
class ArbitraryIcon : Icon {
  private int border;
  public override void draw(){. . .}
  public new int obtainOrigin() {. . .}
}
class Button : RectangularIcon {
 public override void draw(){. . . }
}
class OKButton : Button {
 public override void draw(){. . . }
 public new bool isInside (int p){ . . .}
}
```
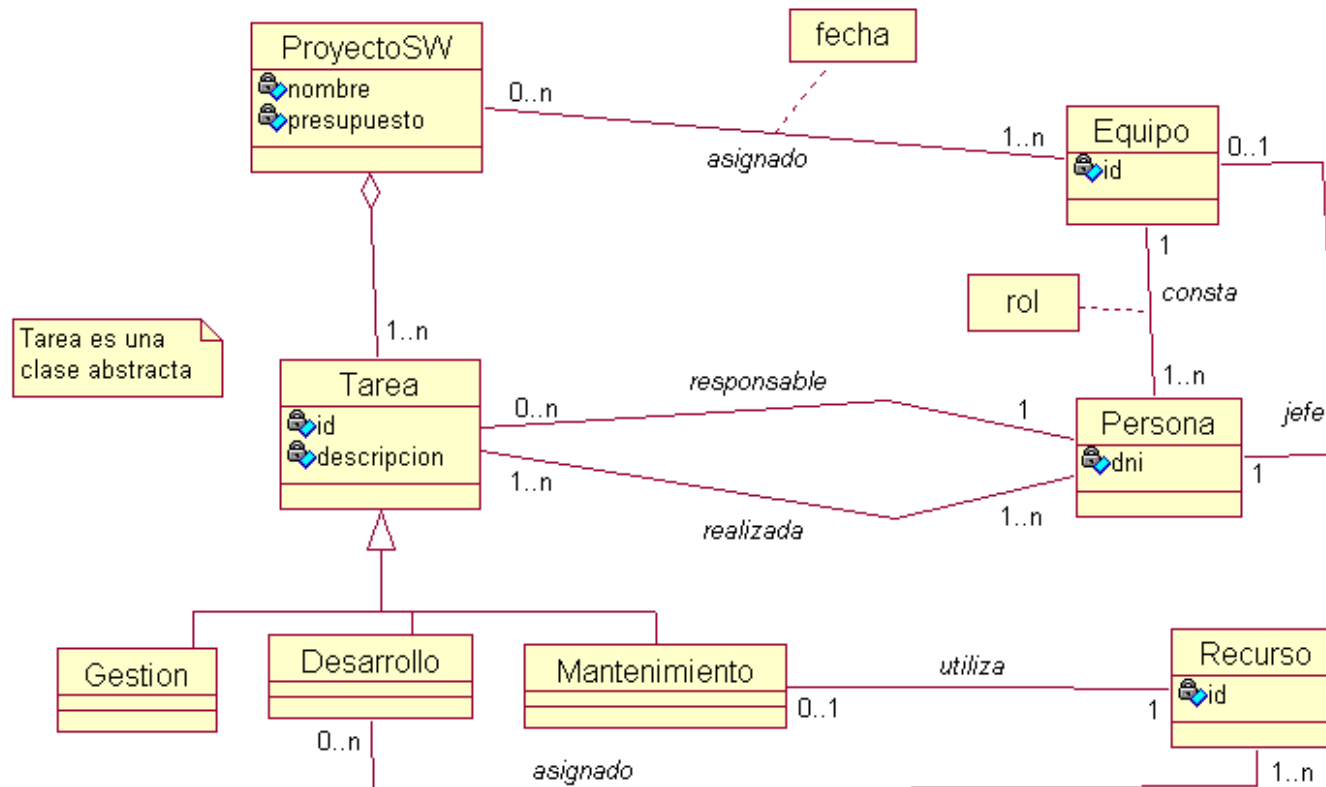
Explain what happens at each instruction
Marked with a number (Indicate whether
A compilation or execution error occurs and
why. Otherwise indicate what method is
 executed).

```
Icon I1 = new RectangularIcon();
Icon I2 = new ArbitraryIcon();
Button B1 = new Button();
Button B2 = new OKButton();
I2.draw();            {1}
I1 = B1;
I1.draw();            {2}
I1.isInside(3);       {3}
B2.draw();            {4}
B2 = B1;
B2.isInside(2);       {5}
I2.obtainOrigin();  {6}
```

3

Constructores

2. For the following system:

a) Declare and implement in C# the constructors for all the designed classes, if needed, so that it is guaranteed the behavior expressed in the model.

b) According to the defined constructors, what would be a correct order of invocation? Call the constructors to illustrate this to create at least one instance of each class and the system ends in a correct and consistent state
.

4

Reuse

3. Many programming languages do not support multiple inheritance and in the conceptual modelling of a system this type of inheritance may be needed. How would you implement the following model?

A
a1 : int
a2 : char
m1()
m2()

B
b1 : char
b2 : int
m1()
s1()

C
c1 : char
m4()