

Programación Orientada a Objetos

Lo abstracto y las interfaces

CEIS

2020-2

Agenda

Lo abstracto

- Universidad

- Otros ejemplos

Interfaz

- Universidad

- Otros ejemplos

Shapes

- Refactorización

- Extensión

- Manipulando

Batalla naval

- Estructura

- Métodos

Agenda

Lo abstracto

Universidad

Otros ejemplos

Interfaz

Universidad

Otros ejemplos

Shapes

Refactorización

Extensión

Manipulando

Batalla naval

Estructura

Métodos

Lo abstracto

Cursos

```
public abstract class Course {  
    // Details omitted.  
    public void enrollStudent(Student s) {  
        enrolledStudents.add(s);  
    }  
    public final void assignInstructor(Professor p) {  
        setInstructor(p);  
    }  
    public void initializeCourse(Professor p, String s, String e) {  
        assignInstructor(p);  
        reserveClassroom();  
        establishCourseSchedule(s, e);  
    }  
    public abstract void establishCourseSchedule(String startDate, String endDate);  
}
```

- ¿Qué se está diciendo? (de la clase, de los métodos)

Reversa

Lo abstracto

Cursos

```
public abstract class Course {  
    // Details omitted.  
    public void enrollStudent(Student s) {  
        enrolledStudents.add(s);  
    }  
    public final void assignInstructor(Professor p) {  
        setInstructor(p);  
    }  
    public void initializeCourse(Professor p, String s, String e) {  
        assignInstructor(p);  
        reserveClassroom();  
        establishCourseSchedule(s, e);  
    }  
    public abstract void establishCourseSchedule(String startDate, String endDate);  
}
```

¿Qué métodos

- ▶ a) deben ser igual para todos?

Lo abstracto

Cursos

```
public abstract class Course {  
    // Details omitted.  
    public void enrollStudent(Student s) {  
        enrolledStudents.add(s);  
    }  
    public final void assignInstructor(Professor p) {  
        setInstructor(p);  
    }  
    public void initializeCourse(Professor p, String s, String e) {  
        assignInstructor(p);  
        reserveClassroom();  
        establishCourseSchedule(s, e);  
    }  
    public abstract void establishCourseSchedule(String startDate, String endDate);  
}
```

¿Qué métodos

- ▶ a) deben ser igual para todos? b) pueden ser igual para todos ?

Lo abstracto

Cursos

```
public abstract class Course {  
    // Details omitted.  
    public void enrollStudent(Student s) {  
        enrolledStudents.add(s);  
    }  
    public final void assignInstructor(Professor p) {  
        setInstructor(p);  
    }  
    public void initializeCourse(Professor p, String s, String e) {  
        assignInstructor(p);  
        reserveClassroom();  
        establishCourseSchedule(s, e);  
    }  
    public abstract void establishCourseSchedule(String startDate, String endDate);  
}
```

¿Qué métodos

- ▶ a) deben ser igual para todos? b) pueden ser igual para todos ?
- ▶ c) son totalmente diferentes para todos ?

Lo abstracto

Cursos

```
public abstract class Course {  
    // Details omitted.  
    public void enrollStudent(Student s) {  
        enrolledStudents.add(s);  
    }  
    public final void assignInstructor(Professor p) {  
        setInstructor(p);  
    }  
    public void initializeCourse(Professor p, String s, String e) {  
        assignInstructor(p);  
        reserveClassroom();  
        establishCourseSchedule(s, e);  
    }  
    public abstract void establishCourseSchedule(String startDate, String endDate);  
}
```

¿Qué métodos

- ▶ a) deben ser igual para todos? b) pueden ser igual para todos ?
- ▶ c) son totalmente diferentes para todos ?
- ▶ d) tienen algunas partes iguales para todos ?

Lo abstracto

Cursos- ¿Crear?

```
Course c = new Course(); // Impossible!
```

Here's the error message:

Course is abstract; cannot be instantiated

No se permite crear un objeto de una clase abstracta

Lo abstracto

Cursos- ¿Crear?

```
Course c = new Course(); // Impossible!
```

Here's the error message:

Course is abstract; cannot be instantiated

No se permite crear un objeto de una clase abstracta

Cursos - ¿Manipular?

```
ArrayList<Course> courses = new ArrayList<Course>();  
// Add a variety of different Course types to the collection.  
courses.add(new LectureCourse());  
courses.add(new LabCourse());  
courses.add(new IndependentStudyCourse());  
// etc.  
for (Course c : courses) {  
    // This next line of code is polymorphic.  
    c.establishCourseSchedule("1/24/2005", "5/10/2005");  
}
```

Se pueden almacenar, representa a sus subclases.

Se pueden usar los métodos, si todas las subclases lo tienen

Agenda

Lo abstracto

Universidad

Otros ejemplos

Interfaz

Universidad

Otros ejemplos

Shapes

Refactorización

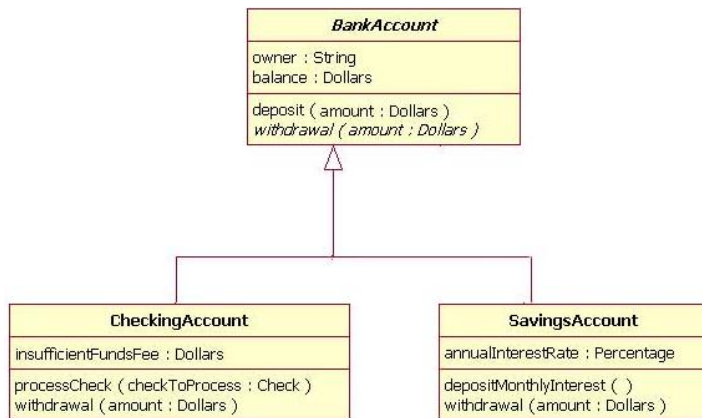
Extensión

Manipulando

Batalla naval

Estructura

Métodos



Manual de referencia. Java.

```
abstract class Point {  
    int x = 1, y = 1;  
    void move(int dx, int dy) {  
        x += dx;  
        y += dy;  
        alert();  
    }  
    abstract void alert();  
}  
abstract class ColoredPoint extends Point {  
    int color;  
}  
  
class SimplePoint extends Point {  
    void alert() { }  
}
```

► ¿Qué se está diciendo?

Reversa

Agenda

Lo abstracto

Universidad

Otros ejemplos

Interfaz

Universidad

Otros ejemplos

Shapes

Refactorización

Extensión

Manipulando

Batalla naval

Estructura

Métodos

Personas y responsabilidades

Para los que investigan

```
import java.time.LocalDate;  
  
public interface Investigator{  
    boolean approveSyllabus(Syllabus s);  
}
```

► ¿Qué se está diciendo ?

Reversa

Personas y responsabilidades

Para los que investigan

```
import java.time.LocalDate;  
  
public interface Investigator{  
    boolean approveSyllabus(Syllabus s);  
}
```

- ▶ ¿Qué se está diciendo ?
Reversa
- ▶ ¿Cómo digo que un estudiante de PHD puede investigar?
¿Qué implica lo anterior?

Personas y responsabilidades

Para los que enseñan

```
import java.time.LocalDate;

public interface Teacher{

    void designedTextbook(TextBook t, Course c);

    Syllabus defineSyllabus(Course c);

    int getHourlyRate();

    default int teachingHourlyRate(){
        return (int)(getHourlyRate()*0.80*(1+(LocalDate.now().getDayOfWeek().getValue()-5 > 0 ? 0.1 : 0)));
    }
}
```

- ¿Qué se está diciendo ?
Reversa

Personas y responsabilidades

Para los que enseñan

```
public class Professor extends Person implements Teacher, Investigator{  
    private static final int HOURLY_RATE = 300000;  
  
    public void designedTextbook(TextBook t, Course c){  
    }  
  
    public Syllabus defineSyllabus(Course c){  
        ...  
    }  
  
    public boolean approveSyllabus(Syllabus s){  
        ...  
    }  
  
    public int getHourlyRate(){  
        return HOURLY_RATE;  
    }  
  
    public int todayClassPayment(int hours){  
        return teachingHourlyRate()*hours;  
    }  
}
```

- ¿Qué se está diciendo ?

Reversa

Agenda

Lo abstracto

Universidad

Otros ejemplos

Interfaz

Universidad

Otros ejemplos

Shapes

Refactorización

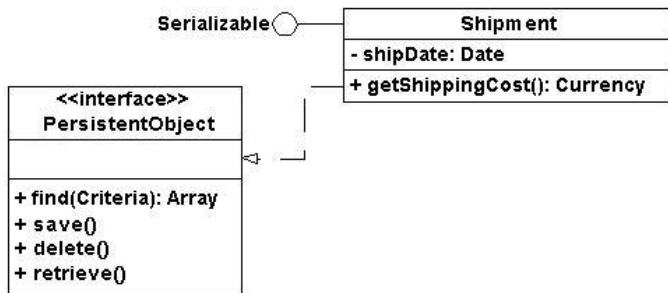
Extensión

Manipulando

Batalla naval

Estructura

Métodos



► ¿Qué se está diciendo?

Manual de referencia. Java.

```
public interface Colorable {
    void setColor(byte r, byte g, byte b);
}
class Point { int x, y; }
class ColoredPoint extends Point implements Colorable {
    byte r, g, b;
    public void setColor(byte rv, byte gv, byte bv) {
        r = rv; g = gv; b = bv;
    }
}
class Test {
    public static void main(String[] args) {
        Point p = new Point();
        ColoredPoint cp = new ColoredPoint();
        p = cp;
        Colorable c = cp;
    }
}
```

► ¿Qué se está diciendo?

Reversa

Agenda

Lo abstracto

Universidad

Otros ejemplos

Interfaz

Universidad

Otros ejemplos

Shapes

Refactorización

Extensión

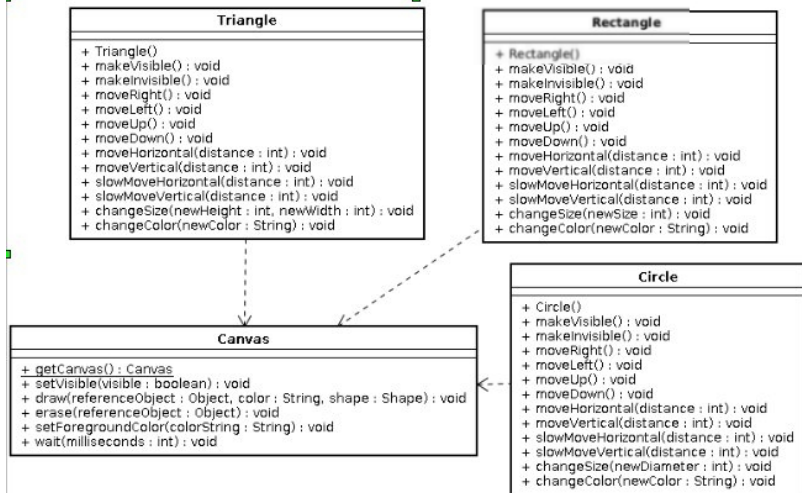
Manipulando

Batalla naval

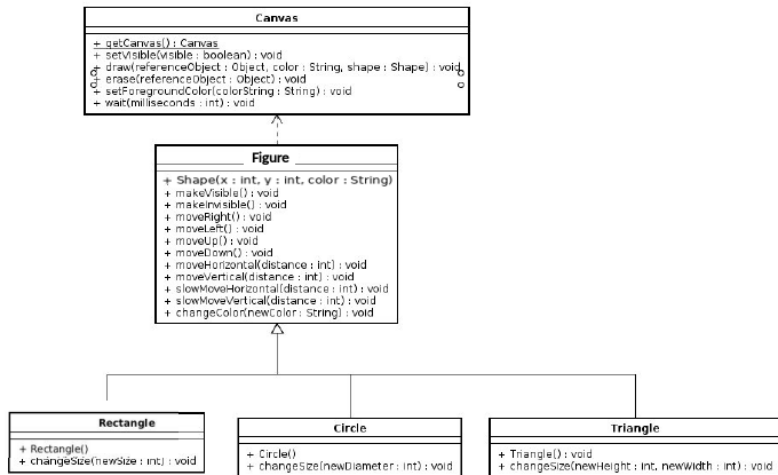
Estructura

Métodos

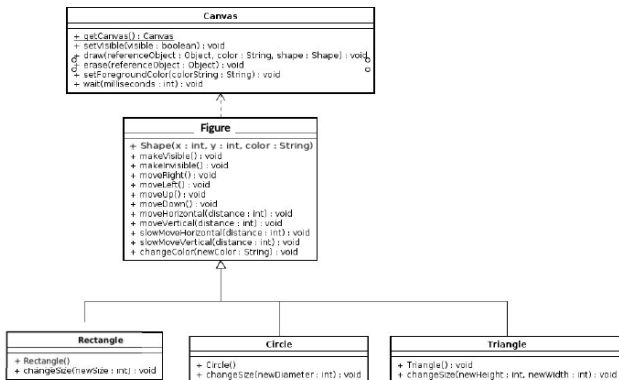
Shapes



Shapes



Shapes

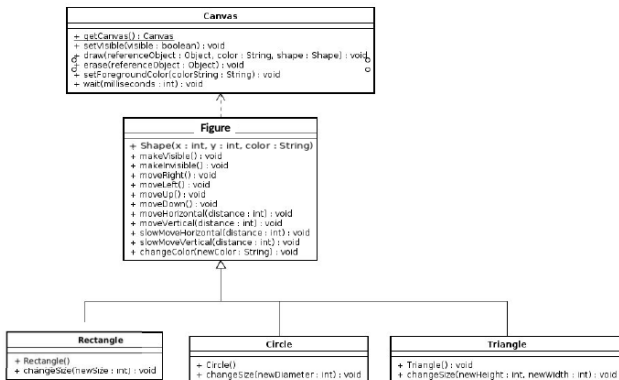


Puliendo

¿Cómo ..

- ▶ podemos impedir que se creen figuras sin sentido?

Shapes

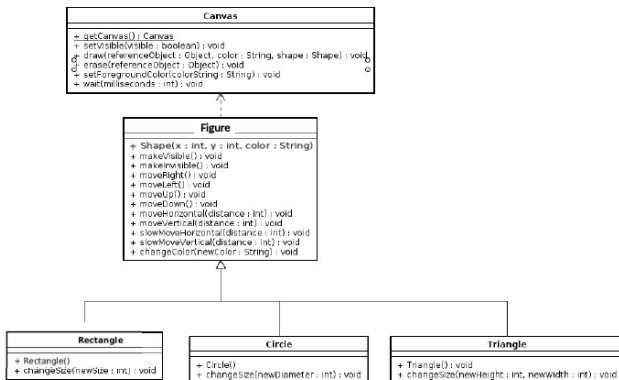


Puliendo

¿Cómo ..

- ▶ podemos impedir que se creen subclases de **Circle**?

Shapes

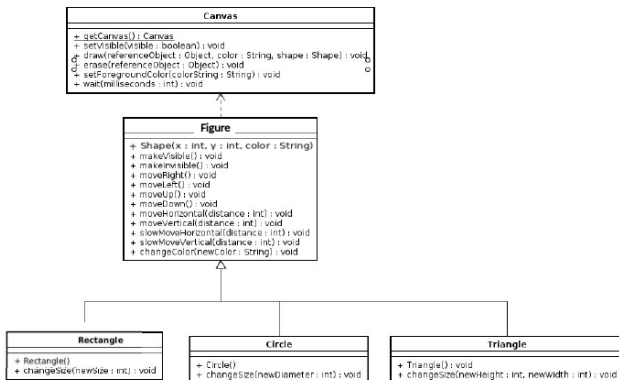


Puliendo

¿Cómo ..

- ▶ podemos exigir que todas las figuras hagan zoom?

Shapes

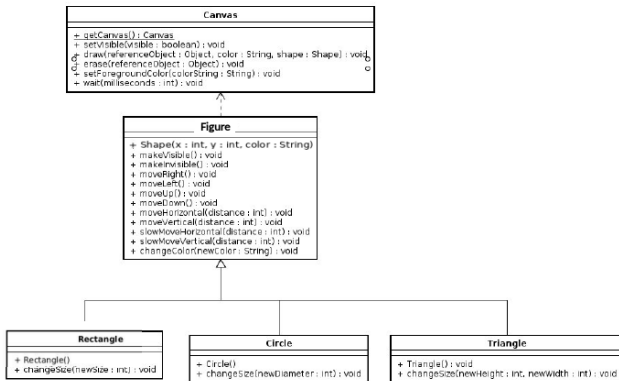


Puliendo

¿Cómo ..

- ▶ podemos impedir que se cambie la forma en que las figuras cambian de color?

Shapes



Puliendo

¿Cómo ..

- ▶ podemos exigir que algunas figuras (por ahora rectángulo y triángulo) retornen cuadrados equivalentes?

Squarable - quadrature

Agenda

Lo abstracto

Universidad

Otros ejemplos

Interfaz

Universidad

Otros ejemplos

Shapes

Refactorización

Extensión

Manipulando

Batalla naval

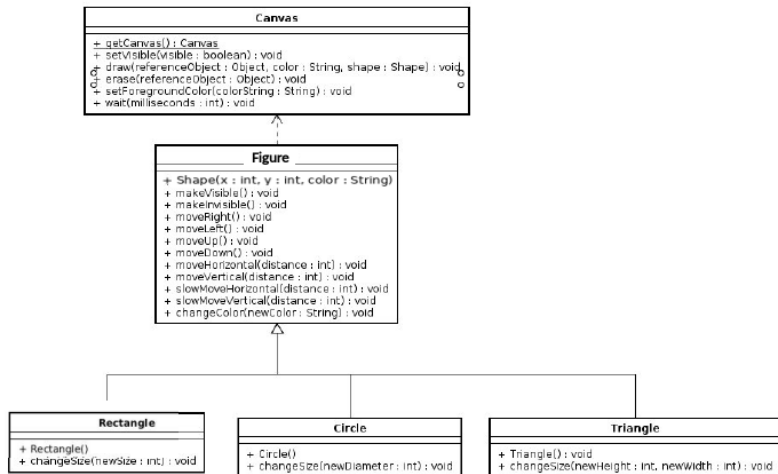
Estructura

Métodos

Shapes

Nueva figura

► Línea



Agenda

Lo abstracto

Universidad

Otros ejemplos

Interfaz

Universidad

Otros ejemplos

Shapes

Refactorización

Extensión

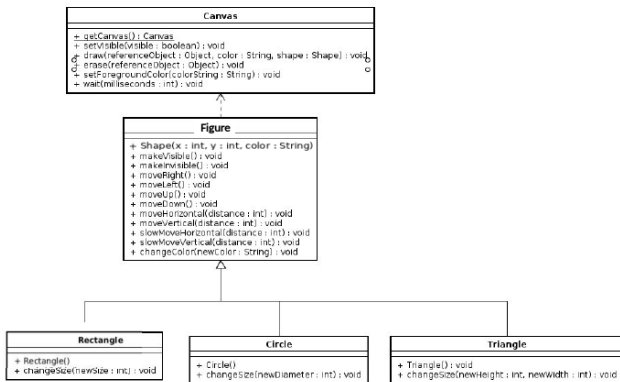
Manipulando

Batalla naval

Estructura

Métodos

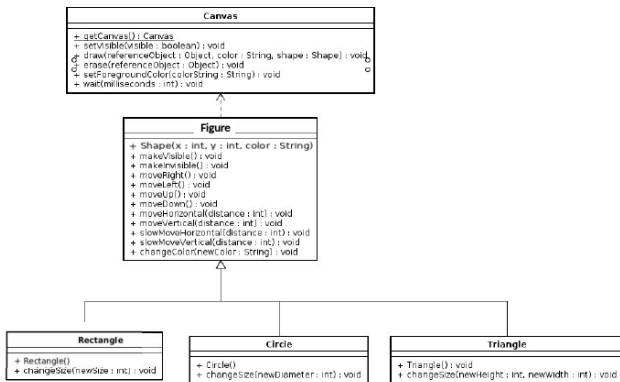
Shapes



ArrayList <Shape> myShapes

- Todas las figuras se mueven a la izquierda

Shapes

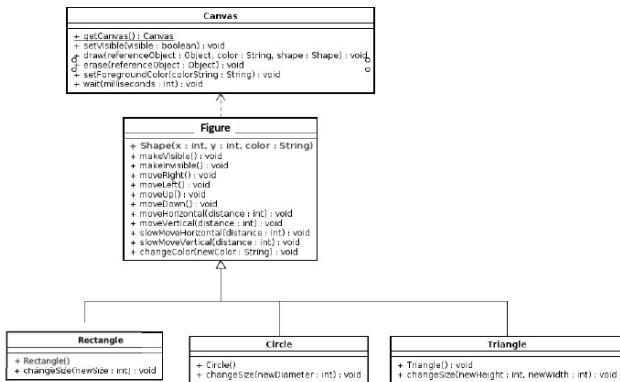


ArrayList <Shape> myShapes

▶ Todos los cuadrados se apagan

instanceof

Shapes



ArrayList <Shape> myShapes

- Un tipo especial de figura se agranda. String special

getClass().getName()

Agenda

Lo abstracto

Universidad

Otros ejemplos

Interfaz

Universidad

Otros ejemplos

Shapes

Refactorización

Extensión

Manipulando

Batalla naval

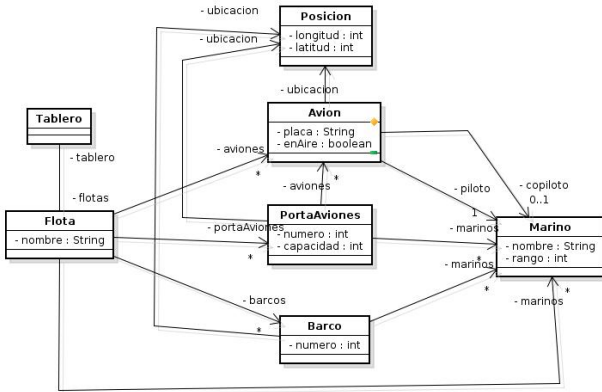
Estructura

Métodos

Batalla naval

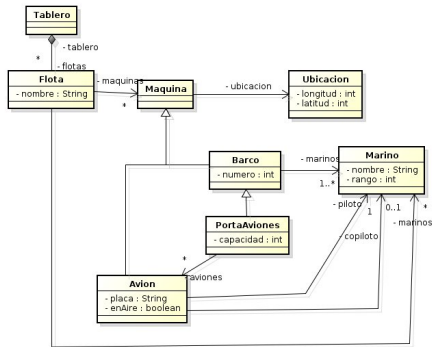
Mejor estructura

- Aprovechando la herencia



Batalla naval

Batalla naval

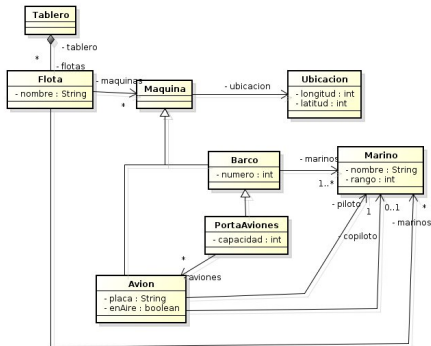


Refactorizando

- ¿Quién debe ser abstracta?

Batalla naval

Batalla naval



Refactorizando

- ▶ ¿Quién debe ser abstracta?
- ▶ ¿Quién podría ser final?

Agenda

Lo abstracto

Universidad

Otros ejemplos

Interfaz

Universidad

Otros ejemplos

Shapes

Refactorización

Extensión

Manipulando

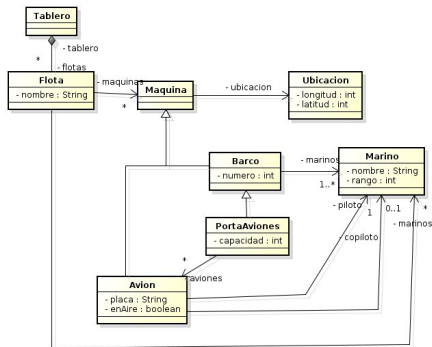
Batalla naval

Estructura

Métodos

Batalla naval

Batalla naval

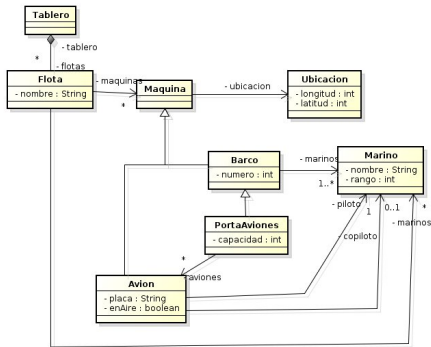


Dos métodos

- Incorporar avance

Batalla naval

Batalla naval



Dos métodos

- ▶ Incorporar avance
- ▶ Incorporar maquinasDebiles