Guía de la asignatura de Sistemas Informáticos Industriales (ADOO)

Carlos Platero Dueñas (C-305)
carlos.platero@upm.es

Contenidos generales

- Dos partes
  - Herramientas y Sistemas operativos (SO) (7-12)
    - Ángel Rodriguez (AR)
  - Análisis y diseño orientado a objetos (ADOO) (1-6)
    - Carlos Platero (CP)
- Planificación
  - 15 semanas
    - 3 horas de SO (M-J) (7-12) (AR)
    - 2 horas de ADOO (V) (1-6) (CP)

Contenidos ADOO

- Programa:
  1. Introducción a la Ingeniería de la Programación.
  2. Recogida y documentos de requisitos
  3. Análisis orientado a objetos
  4. UML: Modelo estructural
  5. UML: Modelo dinámico y de implementación
  6. Diseño orientado a objetos

Evaluación

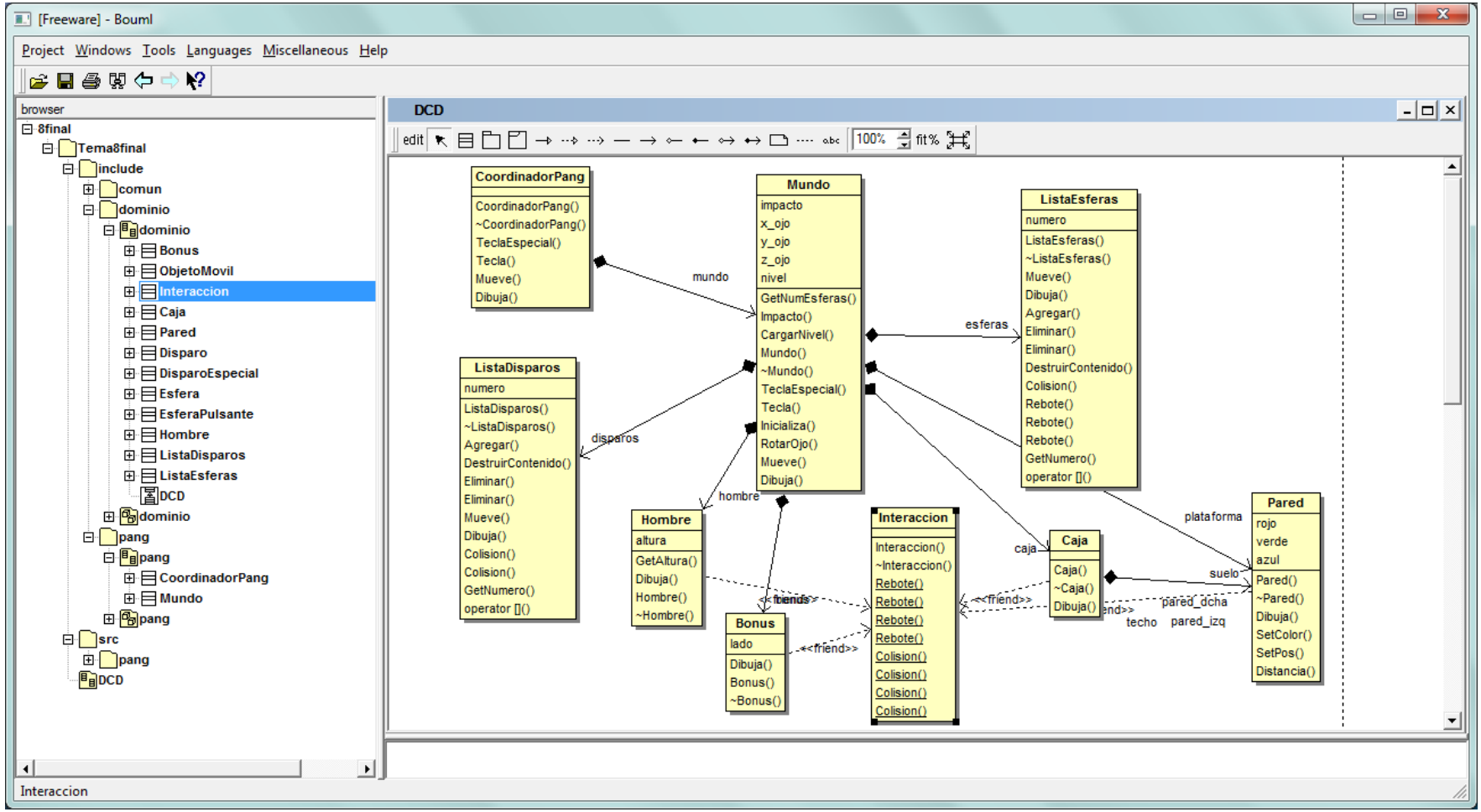
- Prácticas: 25% (obligatorias)
- Evaluación continua 75%:
  - Se realizarán dos pruebas y al menos la suma de ambos deberá ser de 9 o más puntos para su compensación. Con 10 puntos, la parte de teoría-problemas quedará superada definitivamente.
  - Primer examen: 28/10/16
  - Segundo examen: 22/12/16
  - Acuerdo del Consejo de Departamento:
    - "En todas las asignaturas del departamento los alumnos presentados al primer ejercicio de evaluación continua no puedan optar por otro método de evaluación".
- Evaluación no continua
  - Examen final: 75% (horario jefatura de estudios)
- Otros
  - Se conserva el aprobado en la parte práctica y teórica en sucesivas convocatorias.

Material de apoyo

- Bibliografía ADOO
  - *Platero C., Apuntes de ADOO, 2016 (www.ieef.upm.es)*
  - Larman, C., *UML y Patrones. Una introducción al análisis y diseño orientado a objetos y al proceso unificado*, Segunda edición, Pearson Educación 2002.
  - Gamma, E., Helm, R., Jonson, R., Vlissides, J., *Patrones de diseño*, Addison-Wesley, 2003.
- Material en MOODLE (ELAI)
  - Diapositivas
  - Referencias WEB
  - Manuales
- Entornos de programación
  - BoUML
  - Suite de programación: Visual C++, Codeblock,
  - CMake

Herramientas

- BoUML



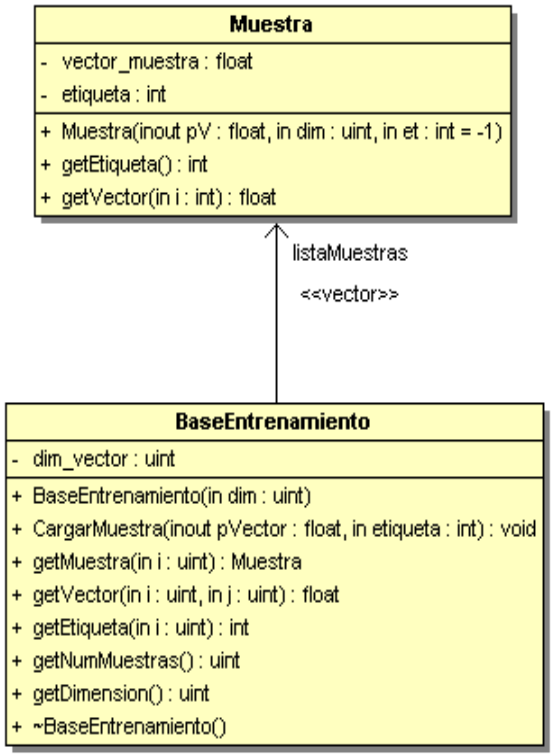
Ejemplo de examen (parte ADOO)

Los clasificadores son parte esencial de muchos programas de Ingeniería. Normalmente, hay un conjunto de muestras de entrenamiento, las cuales tienen definido tanto el vector de características como la etiqueta que se asociada a la clase que le corresponde. Hay muchos tipos de clasificadores: Bayes, redes neuronales, lógica borrosa,… De los clasificadores más simples están los denominados kNN (*k-Nearest Neighbours*). En esta primera versión se va a implementar un clasificador de tipo kNN: ante una nueva muestra, ésta queda clasificada con la etiqueta de la muestra de entrenamiento con menor distancia Euclídea entre vectores. Se pide:

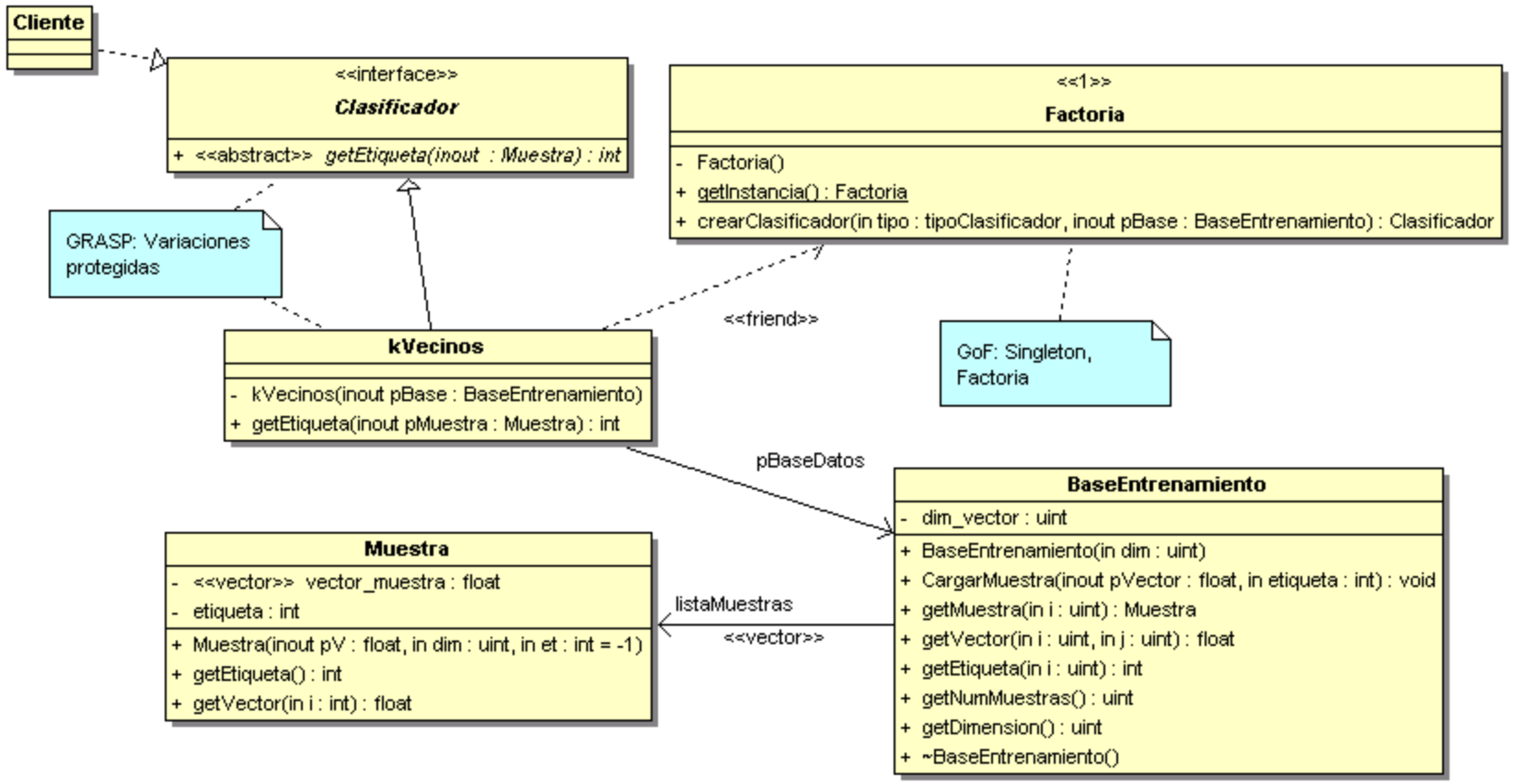
1. Ingeniería Inversa de las clases *Muestra* y *BaseEntrenamiento* (3 puntos).
2. Diagrama de clase de diseño DCD en UML de las clases *Clasificador*, *kVecinos* y *Factoria*. Indique los patrones empleados (3 puntos).
3. Implementación de estas últimas clases en C++ (4 puntos).

Ejemplo de examen (parte ADOO)

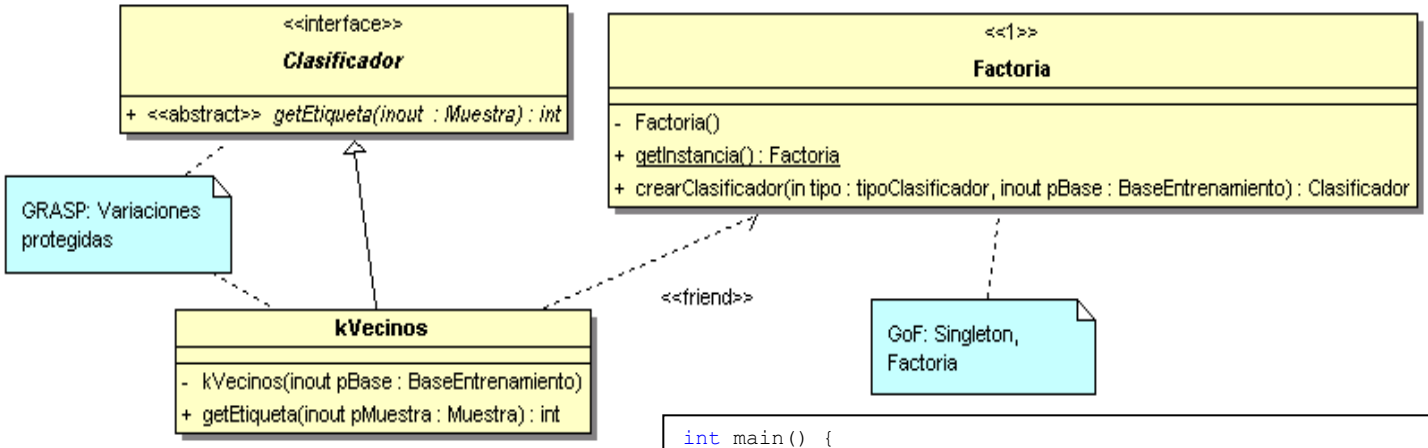
```
#include <vector>
#include <iostream>
using namespace std;
class Muestra {
  vector<float> vector_muestra;
  int etiqueta;
  public:
  Muestra(float *pV,unsigned dim,int et=-1): etiqueta(et){
               for(unsigned i=0;i<dim;i++)
                vector_muestra.push_back(pV[i]);
               }
  int getEtiqueta(){return etiqueta;}
  float getVector(int i) {return vector_muestra[i];}
};

class BaseEntrenamiento{
  vector<Muestra *> listaMuestras;
  unsigned dim_vector;
  public:
  BaseEntrenamiento(unsigned dim):dim_vector(dim){}
  void CargarMuestra(float *pVector,int etiqueta){
       listaMuestras.push_back(new Muestra(pVector,dim_vector,etiqueta));
               }
  Muestra * getMuestra(unsigned i) {return listaMuestras[i];}
  float getVector(unsigned i, unsigned j) {return
listaMuestras[i]->getVector(j);}
  int getEtiqueta(unsigned i) {return
listaMuestras[i]->getEtiqueta();}
  unsigned getNumMuestras() { return listaMuestras.size(); }
  unsigned getDimension() {return dim_vector;}
  ~BaseEntrenamiento() {
    for(unsigned i=0; i<listaMuestras.size();i++)
               delete listaMuestras[i];
     }
};
```



Ejemplo de examen (parte ADOO)



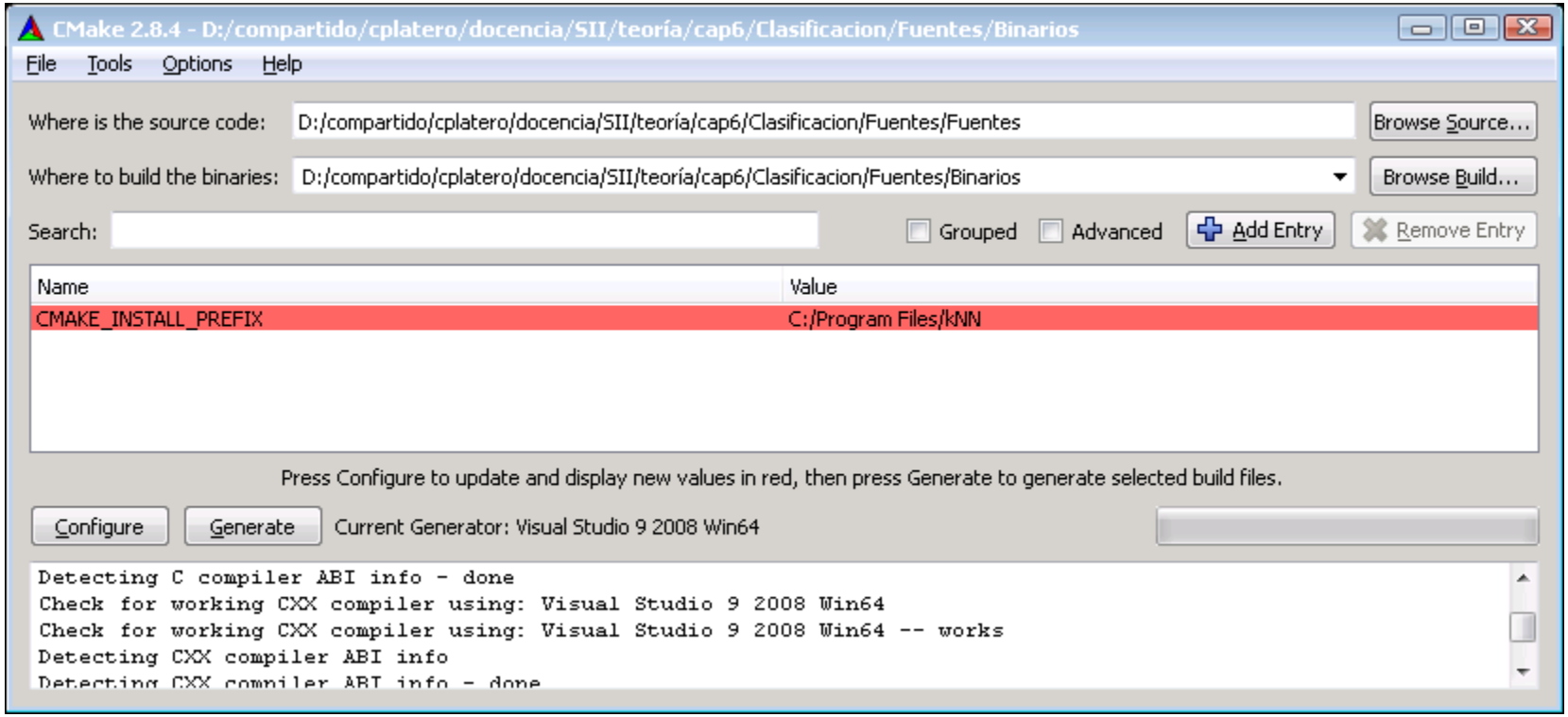
Ejemplo de examen (parte ADOO)




```
int main() {
 unsigned dim = 2;
 unsigned numMuestras = 5;
 float vectores_muestra[5][2] = { {1.0f,1.0f},{0.0f,0.0f},
              {2.0f,2.0f}, {2.0f,3.0f}, {3.0f,2.0f} };
 int etiquetas_muestra[] = {1,1,2,2,2};
 BaseEntrenamiento laBase(dim);

 for(unsigned i=0;i<numMuestras;i++)
  laBase.CargarMuestra(vectores_muestra[i],etiquetas_muestra[i]);

 float vector_nuevo[]={1.0,2.0};
 Muestra laMuestra(vector_nuevo,dim);

 // Clasificar la muestra
 Factoria laFactoria = Factoria::getInstancia();
 Clasificador *pClasificador =
      laFactoria.crearClasificador(kNN,&laBase);
 cout <<"La muestra con vector: "<< laMuestra.getVector(0)<<" "
 << laMuestra.getVector(1) << " tiene la etiqueta: "
 << pClasificador->getEtiqueta(&laMuestra) << endl;

return 0;
}
```

No dependencia del S.O.

- Cmake
  - CMakeLists



No dependencia del S.O.