UNIVERSIDAD POLITÉCNICA DE MADRID
ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA Y DISEÑO INDSUTRIAL

Departamento Electrónica, Automática e Informática Industrial

APELLIDOS

NOMBRE Nº Mat.

Calificación

ASIGNATURA: SISTEMAS INFORMÁTICOS INDUSTRIALES

CURSO 4º GRUPO Octubre 2013

2. Problema de Análisis y Diseño Orientado a Objetos (4 puntos - 25 minutos)

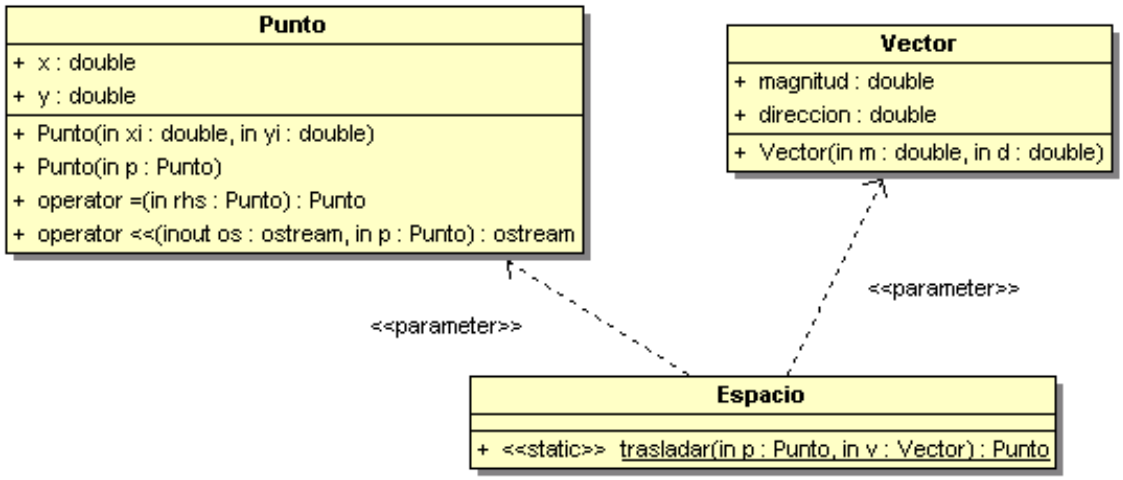
```
#include <iostream>
#include <string>
#include <cmath>
using namespace std;

class Punto {
public:
  double x, y;
  Punto(double xi, double yi) : x(xi), y(yi) {}
  Punto(const Punto& p) :  x(p.x), y(p.y) {}
  Punto& operator=(const Punto& rhs) {
    x = rhs.x;
    y = rhs.y;
    return *this;
  }
  friend ostream&
  operator<<(ostream& os, const Punto& p) {
    return os << "x=" << p.x << " y=" << p.y;
  }
};

class Vector {
public:
  double magnitud, direccion;
  Vector(int m, int d) : magnitud(m), direccion(d) {}
};

class Espacio {
public:
  static Punto trasladar(Punto p, Vector v) {
    p.x += (v.magnitud * cos(v.direccion));
    p.y += (v.magnitud * sin(v.direccion));
    return p;
  }
};

int main() {
  Punto p1(1, 2);
  Punto p2 = Espacio::trasladar(p1, Vector(3, 3.1416/3));
  cout << "p1: " << p1 << " p2: " << p2 << endl;

  return 0;
}
```

Para el código adjuntado se pide:

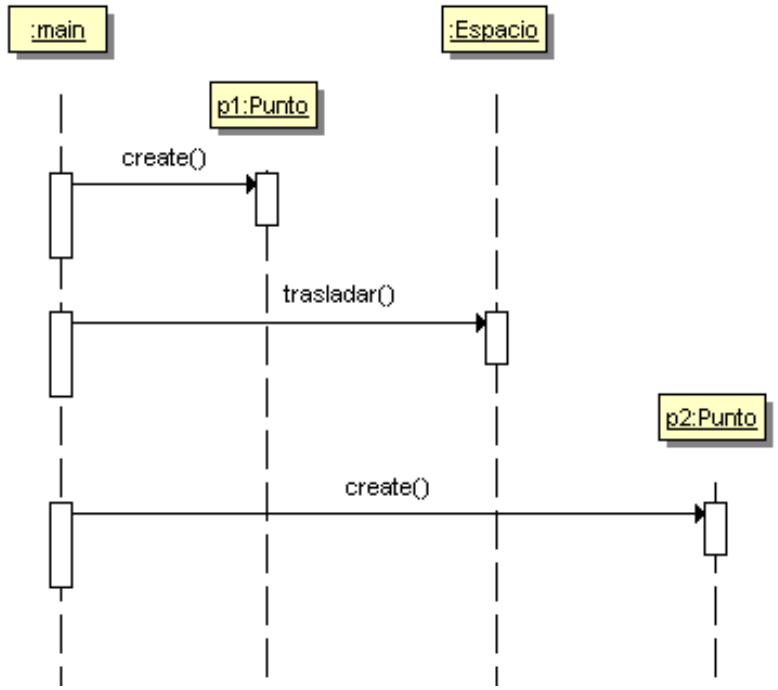
1. Ingeniería inversa: Diagrama de clases.
2. Diagrama de secuencia de la función *main()*.
3. Resultado de su ejecución en la consola.
4. Diseñar e implementar el servicio *rotar( )*, tal que $\begin{pmatrix} x_2 \\ y_2 \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} x_1 \\ y_1 \end{pmatrix}$

Empléese sobre el punto *p3*.

1.

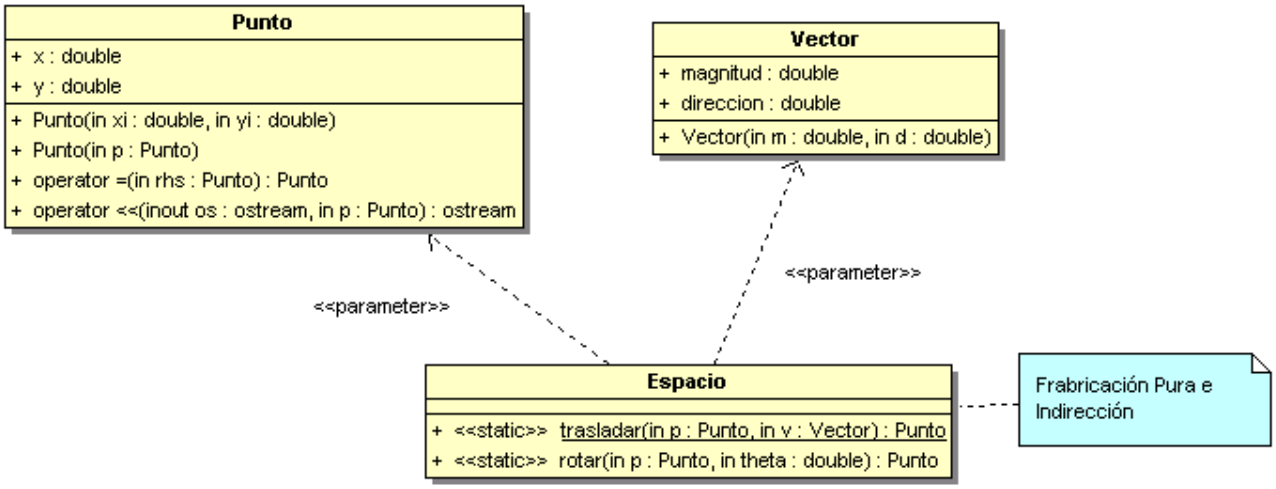


2.



3. p1: x=1 y=2 p2: x=2.5 y=4.6

4. Se ha aplicado Experto de Información en la clase Punto y Vector. Para evitar el acoplamiento entre ambas clases se ha aplicado el patrón Indirección y por tanto una Fabricación Pura con la clase Espacio. El servicio rotar() será responsabilidad de la clase Espacio.



UNIVERSIDAD POLITÉCNICA DE MADRID
ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA Y DISEÑO INDSUTRIAL

Departamento Electrónica, Automática e Informática Industrial

APELLIDOS

NOMBRE Nº Mat.

Calificación

ASIGNATURA: SISTEMAS INFORMÁTICOS INDUSTRIALES

CURSO 4º **GRUPO** **Octubre 2013**

```
class Espacio {
public:
  static Punto trasladar(Punto p, Vector v) {
    p.x += (v.magnitud * cos(v.direccion));
    p.y += (v.magnitud * sin(v.direccion));
    return p;
  }
  static Punto rotar(Punto p, double theta) {
    Punto res(0,0);
    res.x = (p.x * cos(theta)) - (p.y *sin(theta));
    res.y = (p.x * sin(theta)) + (p.y *cos(theta));
    return res;
  }
};
int main() {
  Punto p1(1, 2);
  Punto p2 = Espacio::trasladar(p1, Vector(3, 3.1416/3));
  Punto p3 = Espacio::rotar(p2,3.1416/6);
  cout << "p1: " << p1 << " p2: " << p2 << " p3: " << p3 <<endl;
  return 0;
}
```

UNIVERSIDAD POLITÉCNICA DE MADRID
ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA Y DISEÑO INDSUTRIAL

Departamento Electrónica, Automática e Informática Industrial

APELLIDOS

NOMBRE Nº Mat.

Calificación

ASIGNATURA: SISTEMAS INFORMÁTICOS INDUSTRIALES

CURSO 4º GRUPO Octubre 2013

3. Problema de Gestión de Procesos – Sistemas Operativos (4 puntos - 25 minutos)

Dado el siguiente programa:

```
#include <sys/types.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/wait.h>

#define MAX 4

int main(void){
  int status, n=0, m=0;
  pid_t pid;

  while (n < MAX) {
   pid = fork();
   switch(pid){
    case -1:
      perror("fork"); exit(1);
    case 0:
      if(getpid()%3 == 0) {
        printf ("PID = %d (Padre = %d)\n", getpid(), getppid());
        exit(1);
       }
      else {
        printf ("PID = %d (Padre = %d)\n", getpid(), getppid());
        exit(2);
      }
    default:
      m++;
      pid = wait(&status);
      if(WIFEXITED(status))
        if(WEXITSTATUS(status) == 2) {
           n++;
           printf ("PID finalizado = %d\n", pid);
       }
    }
    printf("m = %d\n", m);
  }
  printf ("n = %d\n", n);
  return 0;
}
```

a. Dibuja el diagrama jerárquico de procesos creados.

b. ¿Qué operaciones hace el proceso padre tras crear el proceso hijo? Explica el código asociado.

c. En el supuesto de que el PID de la Shell desde la que se lanza el programa es el 10.000, ¿qué escribe el programa en la salida estándar si el sistema operativo asigna la siguiente secuencia de PIDs a los procesos hijo: 10.001, 10.002, 10.003, 10.004, 10.005, 10.006, etc.?

d. Explica qué representa la variable `m` al final del programa.

e. ¿Podría en algún caso iterar indefinidamente el bucle while?

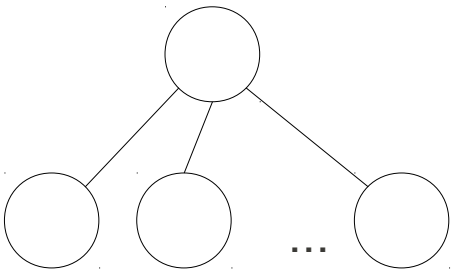
f. Si se elimina el código marcado en negrita y se sustituye por la siguiente línea:

```
m++;
```

Explica cómo afecta esta modificación a la terminación de los procesos.

g. Se desea proteger tanto el proceso inicial como los procesos hijos ante las señales generadas por Ctrl+C. Sobre la versión inicial del código, escribe el código necesario e indica dónde lo incluirías.

a)



b) El padre incrementa *m* y luego espera a la terminación del hijo. Si éste termina con código 2 (PID no divisible por 3), entonces incrementa *n* e imprime el mensaje “PID finalizado ...”.

c)

```
PID = 10002 (Padre = 10001)
m = 1
PID = 10003 (Padre = 10001)
PID finalizado = 10003
m = 2
PID = 10004 (Padre = 10001)
PID finalizado = 10004
m = 3
PID = 10005 (Padre = 10001)
m = 4
PID = 10006 (Padre = 10001)
PID finalizado = 10006
m = 5
PID = 10007 (Padre = 10001)
PID finalizado = 10007
m = 6
n = 4
```

d) *n* es siempre 4 e indica el número de procesos cuyo PID no es divisible por 3.

e) Sí, en el caso en que *n* siempre sea menor que 4, es decir, en el que el sistema operativo no asigne más de 3 PIDs no divisibles por 3 a los sucesivos procesos hijo.

f) El proceso padre no espera por sus hijos, por tanto, se convierten en procesos zombies. La traza de ejecución se modifica al perder el control sobre el orden de finalización de los procesos. Esto afecta al orden en que se imprimen los mensajes por la salida estándar.

g) Se declara la variable:

```
sigset_t mascara;
```

Y antes del while se añade lo siguiente:

```
sigemptyset(&mascara);
sigaddset(&mascara, SIGINT);
sigprocmask(SIG_SETMASK, &mascara, NULL);
```