**PLANTEAMIENTO**
**WHILE-FOR**

Un AEROPUERTO procesa las distintas AEROLINEAS de cada una de estas se conoce: nombre y cantidad de vuelos. De cada VUELO se conoce número del vuelo, cantidad de pasajeros y costo del pasaje. Se pide:

Por Vuelo:

- Monto recaudado

Por Aerolínea:

- Monto recaudado
- Cantidad de vuelos que superan los 3000 $

Por Aeropuerto

- Porcentaje de Vuelos que no superan los 3000 $

FASE 1 Análisis y FASE 2 Diseño (QUEDA PROPUESTA PARA QUE USTED LA REALICE)

FASE 3 IMPLEMENTACIÓN

```
/*
 * Vuelo.h
 */

#ifndef VUELO_H_
#define VUELO_H_
#include <string>
using namespace std;
class Vuelo {
	string numVuelo;
	int cantPasajeros;
	float costPasaje;
public:
	Vuelo();
	void setNumVuelo (string nV);
	void setCantPasajeros (int cP);
	void setCostoPasaje (float costoP);
	string getNumVuelo ();
	int getCantidadPasajeros();
	float getCostoPasaje();
	float calcularMontoRecaudadoXvuelo();


};

#endif /* VUELO_H_ */
```

```
/*
 * Vuelo.cpp
*/

#include "Vuelo.h"

Vuelo::Vuelo() {
	// TODO Auto-generated constructor stub

}

void Vuelo::setNumVuelo (string nV)
{
  numVuelo = nV;
}

void Vuelo::setCantPasajeros (int cP)
{
 cantPasajeros = cP;
}


void Vuelo::setCostoPasaje (float costoP)
{
 costPasaje = costoP;
}

string Vuelo::getNumVuelo ()
{
 return numVuelo;
}

int Vuelo::getCantidadPasajeros()
{
 return cantPasajeros;
}

float Vuelo::getCostoPasaje()
{
 return costPasaje;
}

float Vuelo::calcularMontoRecaudadoXvuelo()
{
 return cantPasajeros* costPasaje;
}
```

```
/*
 * InterfaceVuelo.h
  */

#ifndef INTERFACEVUELO_H_
#define INTERFACEVUELO_H_
#include <string>
#include <iostream>
using namespace std;

class InterfaceVuelo {
public:
	InterfaceVuelo();
	string leerNumVuelo ();
	int leerCantidadPasajeros();
   float leerCostoPasaje ();
	void reporteVuelo(string nV,float mR);


};

#endif



/* INTERFACEVUELO_H_ */
/*
 * InterfaceVuelo.cpp
  */

#include "InterfaceVuelo.h"

InterfaceVuelo::InterfaceVuelo() {
	// TODO Auto-generated constructor stub

}

string InterfaceVuelo::leerNumVuelo ()
{  string nV;
 cout<< "Introduzca Numero del vuelo";
 cin >> nV;
 return nV;
}

int InterfaceVuelo::leerCantidadPasajeros()
{ int cP;
   cout<< "Introduzca cantidad de pasajeros";
   cin >> cP;
return cP;

}
```

```
float InterfaceVuelo::leerCostoPasaje ()
{ float costo;
  cout<< "Introduzca costo del pasaje";
   cin >> costo;
return costo;

}


void InterfaceVuelo::reporteVuelo(string nV,float mR)
{
  cout << "REPORTE DEL VUELO"<< nV<<endl;
  cout << "Monto recaudado"<< mR<<" $"<<endl;

}


/*
 * Aerolinea.h
 *
  */

#ifndef AEROLINEA_H_
#define AEROLINEA_H_
#include "Vuelo.h"
#include <string>
using namespace std;

class Aerolinea {
private:
          string nomb;
          float acumMonto;
          int cantV,contVS, contVNS, contV;


public:
         Aerolinea();
         void setNomb (string n);
    string getNomb();
    void setAcumMonto(float aM);
    float getAcumMonto ();
    void setContVS (int cVS);
    void setContVNS (int cVNS);
    void setContV (int cV);
    void setCantV (int ctV);
    int getContVS ();
    int getContVNS ();
    int getContV ();
    int getCantV ();
    void procesarVuelo (Vuelo vue);


};
```

```
/*
 * Aerolinea.cpp
  */

#include "Aerolinea.h"

Aerolinea::Aerolinea() {
	// TODO Auto-generated constructor stub

}

void Aerolinea::setNomb (string n)
{
  nomb = n;
}

string Aerolinea::getNomb()
{
 return nomb;
}

void Aerolinea::setAcumMonto(float aM)
{
 acumMonto = aM;
}

float Aerolinea::getAcumMonto ()
{
 return acumMonto;
}

void Aerolinea::setContVS (int cVS)
{
 contVS = cVS;
}

void Aerolinea::setContVNS (int cVNS)
{ contVNS = cVNS;
}

void Aerolinea::setContV (int cV)
{
 contV = cV;
}

void Aerolinea::setCantV (int ctV)
{
 cantV = ctV;
}
int Aerolinea::getContVS ()
{
  return contVS;
}
```

```
int Aerolinea::getContVNS ()
{
 return contVNS;
}

int Aerolinea::getContV ()
{
 return contV;
}

int Aerolinea::getCantV ()
{
 return cantV;
}
void Aerolinea::procesarVuelo (Vuelo vue)
{
   acumMonto += vue.calcularMontoRecaudadoXvuelo();

   if (vue.calcularMontoRecaudadoXvuelo()> 3000)
            contVS++;
            else
             contVNS++;

   contV++;
}


/*
 * InterfaceAerolinea.h
  */

#ifndef INTERFACEAEROLINEA_H_
#define INTERFACEAEROLINEA_H_
#include <iostream>
#include <string>
using namespace std;

class InterfaceAerolinea {
public:
         InterfaceAerolinea();
         string leerNombre ();
         int leerCantidadVuelo();
    void reporteAerolinea (float mR, int cVS );
    int leerRespuesta ();
};

#endif /* INTERFACEAEROLINEA_H_ */
```

```
/*
 * InterfaceAerolinea.cpp
*/

#include "InterfaceAerolinea.h"

InterfaceAerolinea::InterfaceAerolinea() {
	// TODO Auto-generated constructor stub

}

string InterfaceAerolinea::leerNombre ()
{string nV;

cout<< "Introduzca Nombre de la Aerolinea";
cin >> nV;
return nV;

}
void InterfaceAerolinea::reporteAerolinea (float mR, int cVS )
{ cout<< endl;
  cout<< "REPORTE AEROLINEA"<<endl;
  cout << "Monto recaudado"<< mR << "$"<<endl;
  cout << "Cantidad de vuelos que superan los 3000 $"<< cVS <<endl;
  cout<< endl;
}
int InterfaceAerolinea::leerCantidadVuelo()
{ int ctV;
  cout<< "Introduzca la cantidad de vuelos de la Aerolinea";
  cin >> ctV;
  return ctV;
}
int InterfaceAerolinea::leerRespuesta ()
{int resp;
cout<< "Desea procesar otra aerolinea 1 Si 2 No";
cin >> resp;
return resp;

}
```

```
/*
 * Aeropuerto.h
*/

#ifndef AEROPUERTO_H_
#define AEROPUERTO_H_
#include "Aerolinea.h"

class Aeropuerto {
private:
	int acumVuelosNS, acumVuelos;

public:
	Aeropuerto();
	void setAcumVuelosNS(int aVNS);
	void setAcumVuelos(int aV);
	int getAcumVuelosNS ();
	int getAcumVuelos ();
	void procesarAerolinea(Aerolinea aer);
	float calcularPorcentaje ();

};

#endif /* AEROPUERTO_H_ */

/*
 * Aeropuerto.cpp
*/

#include "Aeropuerto.h"

Aeropuerto::Aeropuerto() {
	acumVuelos = 0;
	acumVuelosNS = 0;
}

void Aeropuerto::setAcumVuelosNS(int aVNS)
{
  acumVuelosNS = aVNS;
}
void Aeropuerto::setAcumVuelos(int aV)
{
 acumVuelos = aV;
}
int Aeropuerto::getAcumVuelosNS ()
{
 return acumVuelosNS;
}
int Aeropuerto::getAcumVuelos ()
{
 return acumVuelos;
}
```

```
void Aeropuerto::procesarAerolinea(Aerolinea aer)
{
  acumVuelosNS+= aer.getContVNS();
  acumVuelos += aer.getContV();

}

float Aeropuerto::calcularPorcentaje ()
{
 return acumVuelosNS*100 /acumVuelos;
}



/*
 * InterfaceAeropuerto.h
  */

#ifndef INTERFACEAEROPUERTO_H_
#define INTERFACEAEROPUERTO_H_
#include <iostream>
using namespace std;
class InterfaceAeropuerto {
public:
	InterfaceAeropuerto();
	void reporteAerpuerto(float p);
};

#endif /* INTERFACEAEROPUERTO_H_ */



/*
 * InterfaceAeropuerto.cpp
  */

#include "InterfaceAeropuerto.h"

InterfaceAeropuerto::InterfaceAeropuerto() {
	// TODO Auto-generated constructor stub

}

void InterfaceAeropuerto::reporteAerpuerto(float p)
{
	  cout<< endl;
	  cout<< "REPORTE AEROPUERTO"<<endl;
	  cout << "Porcentaje de Vuelos que no superan los 3000 $"<< p << "%"<<endl;

	  cout<< endl;
}
```

```
/*
 * principal.cpp
*/
#include "Vuelo.h"
#include "InterfaceVuelo.h"
#include "Aerolinea.h"
#include "InterfaceAerolinea.h"
#include "Aeropuerto.h"
#include "InterfaceAeropuerto.h"

int main ()
{ Vuelo vue;
  InterfaceVuelo intVue;
  Aerolinea aer;
  InterfaceAerolinea intAer;
  Aeropuerto aerop;
  InterfaceAeropuerto intAerop;

  string nV, nA;
  int ctV,cP,  opc = 1;
  float costoP;

while (opc==1)
{
  nA =intAer.leerNombre();
  aer.setNomb(nA);
  ctV = intAer.leerCantidadVuelo();
  aer.setCantV(ctV);

  aer.setAcumMonto(0.0);
  aer.setContV(0);
  aer.setContVNS(0);
  aer.setContVS(0);


  for (int i=0; i< aer.getCantV(); i++)
  {
        nV = intVue.leerNumVuelo();
        vue.setNumVuelo(nV);

        cP = intVue.leerCantidadPasajeros();
        vue.setCantPasajeros(cP);

        costoP = intVue.leerCostoPasaje();
        vue.setCostoPasaje(costoP);

        intVue.reporteVuelo(vue.getNumVuelo(), vue.calcularMontoRecaudadoXvuelo());

        aer.procesarVuelo(vue);

  }
```

```
   intAer.reporteAerolinea(aer.getAcumMonto(),aer.getContVS());
   aerop.procesarAerolinea(aer);
   opc = intAer.leerRespuesta();
}
   intAerop.reporteAerpuerto(aerop.calcularPorcentaje());

return 0;
}
```