Pauta control 2 CC10B
2007-01

Fórmula para el cálculo de la nota: $Nota(pts) = 1 + \frac{pts}{7}$

Alcance: las soluciones presentadas en esta pauta son sólo referenciales.

Resumen de puntaje

P1: 12.0. Jerarquía de clases 3.0, Departamento 3.0, departamento circular 3.0, departamento rectangular 3.0.

P2: 24.0. Constructor 1.0, agregar 4.0, eliminarDepartamentosVendidos 4.0, comprar 2.0, imprimirDepartamentos 6.0, contarDepartamentos 3.0, metrosCuadradosPromedioCirculares 4.0.

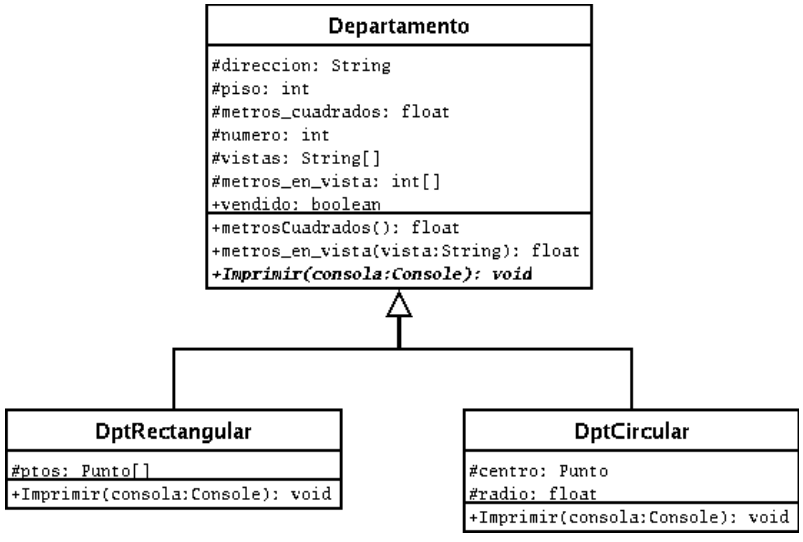
P3: 8.0. Ordenar 8.0.

P4: 6.0. Empresa con 500 departamentos 1.0, agregar departamento circular 1.0 y rcctangular 1.0, imprimir los departamentos con 100 m^2 y vista norte 1.0, comprar un departamento de los agregados 1.0 y eliminar los departamentos vendidos 1.0.

Los departamentos 12 puntos

Se pide:

- Diseñar la jerarquía de clases (sin agregar nuevos métodos). 3 ptos.
- Implementar los métodos especificados. 3 ptos cada departamento bien implementado.



(Se espera esta respuesta, al menos en cuanto a forma)

```
abstract public class Departamento {     3 pts

 protected String direccion;
 protected int piso;
 protected float metros_cuadrados;
 protected int numero;
 protected String[] vistas;
 protected float[] metros_en_vista;
 protected boolean vendido;

 public Departamento(int numero,String direccion,int piso,float
area_piso,String[] vistas,int[] metros_en_vista) {
  numero=numero;
  direccion=direccion;
  metros_cuadrados=area_piso;
  piso=piso;

  //aún no está vendido
  vendido=false;

  //construcción de los arreglos internos
  this.vistas=new String[4];
  this.metros_en_vista=new int[4];
  for(int i=0;i<4;i++) {
   this.vistas[i]=vista[i];
   this.metros_en_vista[i]=metros_en_vista;
  }
 }

 public float metrosCuadrados() {
  return metros_cuadrados;
 }

 public float metros_en_Vista(String vista) {
  //se busca la vista y se retorna inmediatamente
  for(int k=0;k<4;k++) {
    if (vistas[k].equals(vista)) {
      return metros_en_vista[k];
    }
  }
 }

 //Imprimir será escrito en las subclases
 abstract public void Imprimir(Console consola);

}
```

```
public class DptRectangular extends Departamento {   3 pts

 protected Punto[] ptos;

 public DptRectangular(int numero,String direccion,int piso,float
area_piso,String[] vistas,int[] metros_en_vista,Punto[] esquinas) {
  //construcción a través del padre
  super(numero,direccion,piso,area_piso,vistas,metros_en_vista);

  //construcción del arreglo de puntos
  ptos=new Punto[4];
  for(int i=0;i<4;i++) {
    ptos[i]=new Punto(esquinas[i].getX(),esquinas[i].getY());
  }
 }

 public void Imprimir(Console consola) {
  //La información básica del departamento
  consola.println("Departamento "+numero+": Direccion " + dirección +
         ", piso " + piso + ", metros cuadrados " + metros_cuadrados + ".");

  //La forma del departamento y las esquinas
  consola.println("Forma rectangular. Esquinas: ");
  for(int k=0;k<3;k++) {
    consola.print("(" + ptos[k].getX() + "," + ptos[k].getY() +"), " );
  }
  consola.println("(" + ptos[3].getX() + "," + ptos[3].getY() +")." );

  //Se entrega la información de las vistas
  consola.print("Vistas: ");
  for(int k=0;k<4;k++) {
    consola.print(vistas[k] + " " + metros_en_vista[k] + " m2 ");
  }
  consola.println();

  //Por último, se indica su estado de venta
  if (vendido) {
    consola.println("Este departamento ha sido vendido.");
  }else {
    consola.println("Este departamento está en venta.");
  }
 }

}
```

```
public class DptCircular extends Departamento {      3 pts

  protected Punto centro;
  float radio;

 public DptRectangular(int numero,String direccion,int piso,float
area_piso,String[] vistas,int[] metros_en_vista,Punto centro,float radio) {
   //construcción a través del padre
   super(numero,direccion,piso,area_piso,vistas,metros_en_vista);

   //información del círculo
   this.centro=new Punto(centro.getX(),centro.getY());
   this.radio=radio;
 }

 public void Imprimir(Console consola) {
   //La información básica del departamento
   consola.println("Departamento "+numero+": Direccion " + dirección +
          ", piso " + piso + ", metros cuadrados " + metros_cuadrados + ".");

   //La forma del departamento y su círculo
   consola.println("Forma circular. Centro: (" + centro.getX() + "," +
          centro.getY() +"), radio: " + radio + " m2");

   //Se entrega la información de las vistas
   consola.print("Vistas: ");
   for(int k=0;k<4;k++) {
     consola.print(vistas[k] + " " + metros_en_vista[k] + " m2 ");
   }
   consola.println();

   //Por último, se indica su estado de venta
   if (vendido) {
     consola.println("Este departamento ha sido vendido.");
   } else {
     consola.println("Este departamento está en venta.");
   }
 }

}
```

Empresa constructora 24 puntos
Se pide implementar la clase EmpresaConstructora, agregando los métodos necesarios en los departamentos.

Primero se aumentará Departamento algunos métodos para acceder al contenido:

```
abstract public class Departamento {

 ...

 public void vender() {
   vendido=true;
 }

 public boolean vendido() {
   return vendido;
 }

 public String vistaPrincipal() {
   int max=metros_en_vista[0], pos=0;
   for(int i=1;i<4;i++)
     if (metros_en_vista[i]>max) {
       max=metros_en_vista[i];
       pos=i;
     }
   return vistas[pos];
 }

 public String direccion() {
   return direccion;
 }

 public int numero() {
   return numero;
 }

 public int piso() {
   return piso;
 }

}
```

Ahora se implementa EmpresaConstructora:

```
public class EmpresaConstructora {

 private Departamento[] departamentos;
 private int numero_actual_de_departamentos;


 // 1.0 punto
 public EmpresaConstructora(int numero_maximo_departamentos) {
   departamentos=new Departamento[numero_maximo_departamentos];
   numero_actual_de_departamentos=0;
 }


 // 4.0 puntos
 public void agregar(int tipo, String direccion, float metros_cuadrados, int
piso, int numero, String[] vistas, float[] metros_vista, Punto[] puntos, float
radio) {
   int N=numero_actual_de_departamentos;
   if (tipo==0)
     departamentos[N]=new DptRectangular(numero, direccion, piso,
                 metros_cuadrados, vistas, metros_vista, puntos);
   else
     departamentos[N]=new DptCircular(numero, direccion, piso,
                 metros_cuadrados, vistas, metros_vista, puntos[0], radio);
   numero_actual_de_departamentos++;
   reordenar();
 }


 // 4.0 puntos
 public void eliminarDepartamentosVendidos() {
   //la estrategia es construir un nuevo arreglo sólo con los departamentos
   //sin vender... es fácil ver que no es necesario reordenar
   Departamento[] alter=new Departamento[departamentos.length];
   int k=0;
   for(int i=0;i<numero_actual_de_departamentos;i++) {
     if (!departamentos[i].vendido()) {
       alter[k]=departamentos[i];
       k++;
     }
   }
   departamentos=alter;
   numero_actual_de_departamentos=k;
 }
```

```
// 2.0 puntos
public Departamento comprar(String direccion,int numero,int piso) {
  Departamento dpto;
  for(int i=0;i<numero_actual_de_departamentos;i++) {
    if ( departamentos[i].direccion().equals( direccion )
         && departamentos[i].numero() == numero
         && departamentos[i].piso() == piso ) {
      dpto=departamentos[i];
      break;
    }
  }
  dpto.vender();
  return dpto;
}


// 6.0 puntos
public void imprimirDepartamentos(Console consola, float superficie, String
vista) {
  //Es OBLIGATORIO usar búsqueda binaria y es lo primero a realizar
  int p1=0, p2=numero_actual_de_departamentos-1;
  for(;p2>p1;) {
    int k=(p1+p2)/2;
    if (departamentos[k].metrosCuadrados()>=superficie) {
      p1=k;
      if (departamentos[k].metrosCuadrados()==superficie)
        break;
    } else {
      p2=k;
    }
  }
  //Hay que retroceder para recorrer los departamentos con igual superficie
  for(;p1>=0 && departamentos[p1].metrosCuadrados()==superficie;p1--) ;
  //Corrección en caso de que departamentos[p1] no cumpla con superficie
  if (departamentos[p1].metrosCuadrados()<superficie)
    p1++;
  //Ahora se imprimen los departamentos indicados
  //Si no hay departamentos que cumplan, no pasará nada
  for(;p1<numero_actual_de_departamentos &&
              departamentos[p1].metrosCuadrados()==superficie; p1++) {
    if (departamentos[p1].vistaPrincipal().equals(vista)) {
      departamentos[p1].Imprimir(consola);
    }
  }
}
```

```
// 3.0 puntos
public int contarDepartamentos(String vista,float min_metros) {
  //Acá no se exigía búsqueda binaria
  int num=0;
  for(int i=0;i<numero_actual_de_departamentos;i++)
    if (departamentos[i].vistaPrincipal().equals(vista))
      num++;
  return num;
}


// 4.0 puntos
public float metrosCuadradoPromedioCirculares() {
  //Hay que sumar todas las áreas y dividirlas por los circulares
  float area=0;
  int num=0;
  for(int i=0;i<numero_actual_de_departamentos;i++) {
    if (departamentos[i] instanceOf DptCircular) {
      area+=departamentos[i].metrosCuadrados();
      num++;
    }
  }
  return area/num;
}


//lo usé en otro lado
//reordenar ordena cuando un nuevo elemento es añadido
//sólo es necesaria una pasada de la burbuja, de final a inicio
private void reordenar() {
  for(int i=numero_actual_de_departamentos-1;i>0;i--) {
    if (departamentos[i].metrosCuadrados()<
              departamentos[i-1].metrosCuadrados()) {
      Departamento D=departamentos[i];
      departamentos[i]=departamentos[i-1];
      departamentos[i-1]=D;
    }
  }
}

}
```

Ordenamiento por vista principal 8 puntos

Se pide implementar un método para EmpresaConstructora que ordene por vista principal (la de más metros) por orden Este, Norte, Oeste y Sur. Y dentro de cada orden principal, ordenar por los metros.

Se usará burbuja debido a que no se especifica qué algoritmo de ordenamiento usar.

```
public class EmpresaConstructora{

 ...

 public void ordenarVista() {
   boolean salir=false;
   while(!salir) {
     salir=true;
     for(int i=numero_actual_de_departamentos-1;i>0;i--) {
       if (vistaXmenorY(departamentos[i],departamentos[i-1])) {
         Departamento D=departamentos[i];
         departamentos[i]=departamentos[i-1];
         departamentos[i-1]=D;
         salir=false;
       } //if
     } //for
   } //while
 }

 private boolean vistaXmenorY(Departamento X,Departamento Y) {
   if (X.vistaPrincipal().compareTo(Y.vistaPrincipal())<0)
     return true;
   return X.metros_en_vista(X.vistaPrincipal()) <
                Y.metros_en_vista(Y.vistaPrincipal());
 }

}
```

Clase Ventas 6 puntos

Se pide construir una clase Ventas para probar lo implementado. En particular:

- Crear una EmpresaConstructora con capacidad de 500 departamentos. 1 pto.
- Agregar un departamento circular y uno rectangular. 2 ptos.
- Imprimir todos los metros que tengan 100 metros cuadrados y vista Norte. 1 pto.
- Comprar un departamento que agregó. 1 pto.
- Eliminar los departamentos vendidos. 1 pto.

```
public class Ventas{

  public static void main(String[] A) {
    Console C=new Console();
    EmpresaConstructora ec=new EmpresaConstructora(500);
    String[] vistas={"Norte","Sur","Este","Oeste"};
    int[] mcirc={1,2,3,4}, mrect={2,3,4,5};
    Puntos[] esq=new Puntos[4];
    esq[0]=new Punto(0,0);
    esq[1]=new Punto(0,1);
    esq[2]=new Punto(1,1);
    esq[3]=new Punto(1,0);
    ec.agregar(1,"Lejos",2,1,1,vistas,mcirc,esq,1);
    ec.agregar(0,"Cerca",2,1,1,vistas,mrect,esq,1);
    ec.imprimirDepartamentos(C,100,"Norte");
    ec.comprar("Lejos",1,1);
    ec.eliminarDepartamentosVendidos();
  }

}
```