Projecte de Programació

Ejemplo Arquitectura 3 capas Mantenimiento Genérico

Ejemplo Arquitectura 3 capas

Si hiciéramos el mantenimiento de cualquier otro objeto, gran parte de la lógica del ejemplo anterior se mantendría

Se trata de extender el ejemplo anterior al mantenimiento de otras clases que tengan en común:

- Las mismas funcionalidades (Alta, Baja, Consulta)
- Sus objetos se guardan en el mismo contenedor (*TreeMap*)
- Sus objetos tienen un identificador del mismo tipo

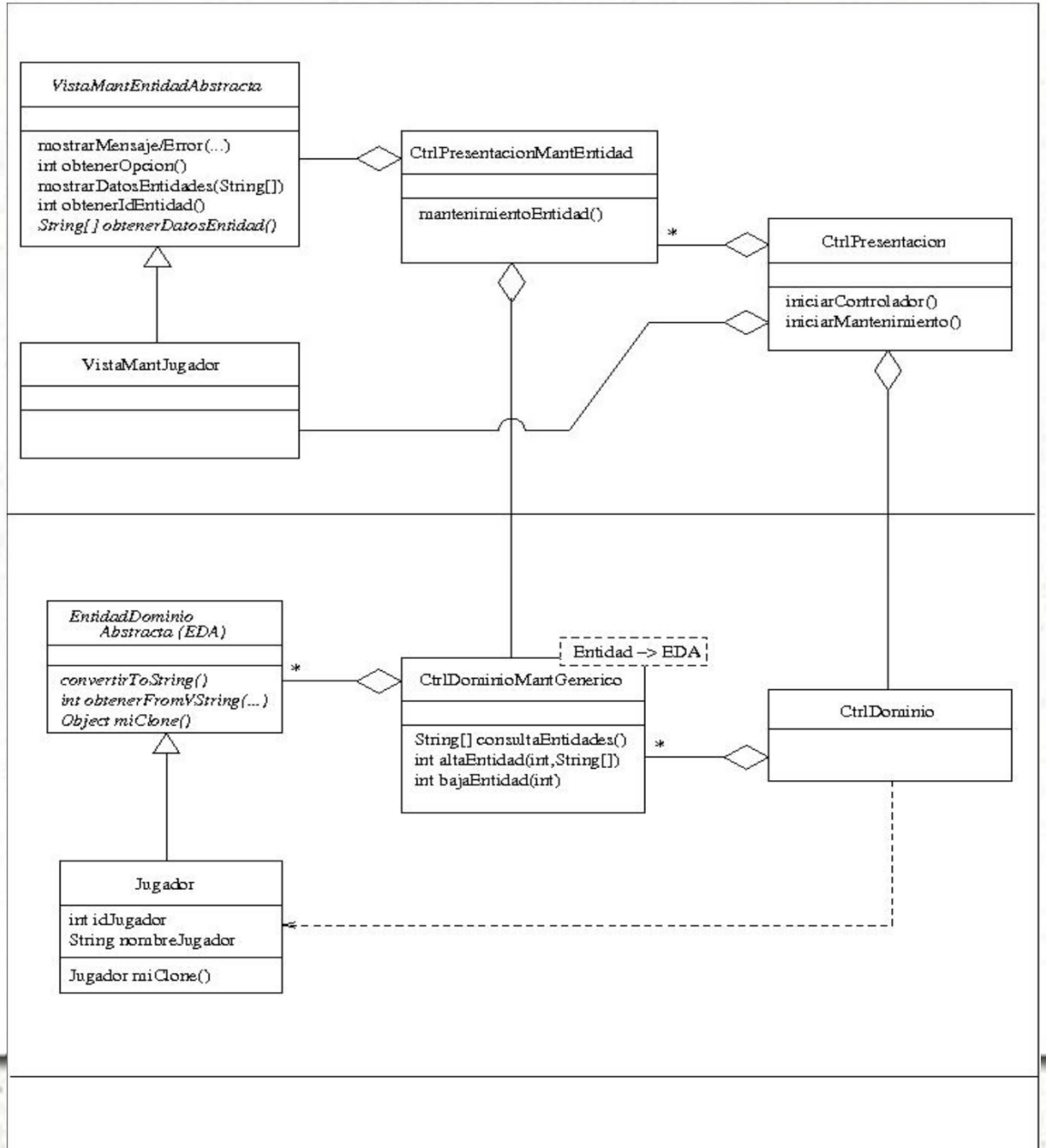
La clase de objetos a mantener será un parámetro de las clases que implementen un **mantenimiento genérico**

Ejemplo Arquitectura 3 capas

Cambios en la Capa de Dominio:

- CtrlDominioMantGenérico, clase parametrizada, con genericidad **restringida**: el parámetro genérico ha de heredar de una nueva clase EntidadDominioAbstracta
- Aparece un nuevo atributo instancia del parámetro genérico, por no poderse llamar a la constructora del parámetro
- Sólo cambia el código de los métodos que accedían a métodos particulares de la clase Jugador. Se han de definir 3 nuevos métodos abstractos en EntidadDominioAbstracta:
  - *String convertirToString()* para la consulta
  - *int obtenerFromVectorString (int, Vector<String>)* para el alta
  - *Object miClone()* para copiar la instancia recibida
- Resto del código, es completamente generalizable y sólo sufre cambios cosméticos (sustituir "Jugador" por "Entidad")

Ejemplo Arquitectura 3 capas

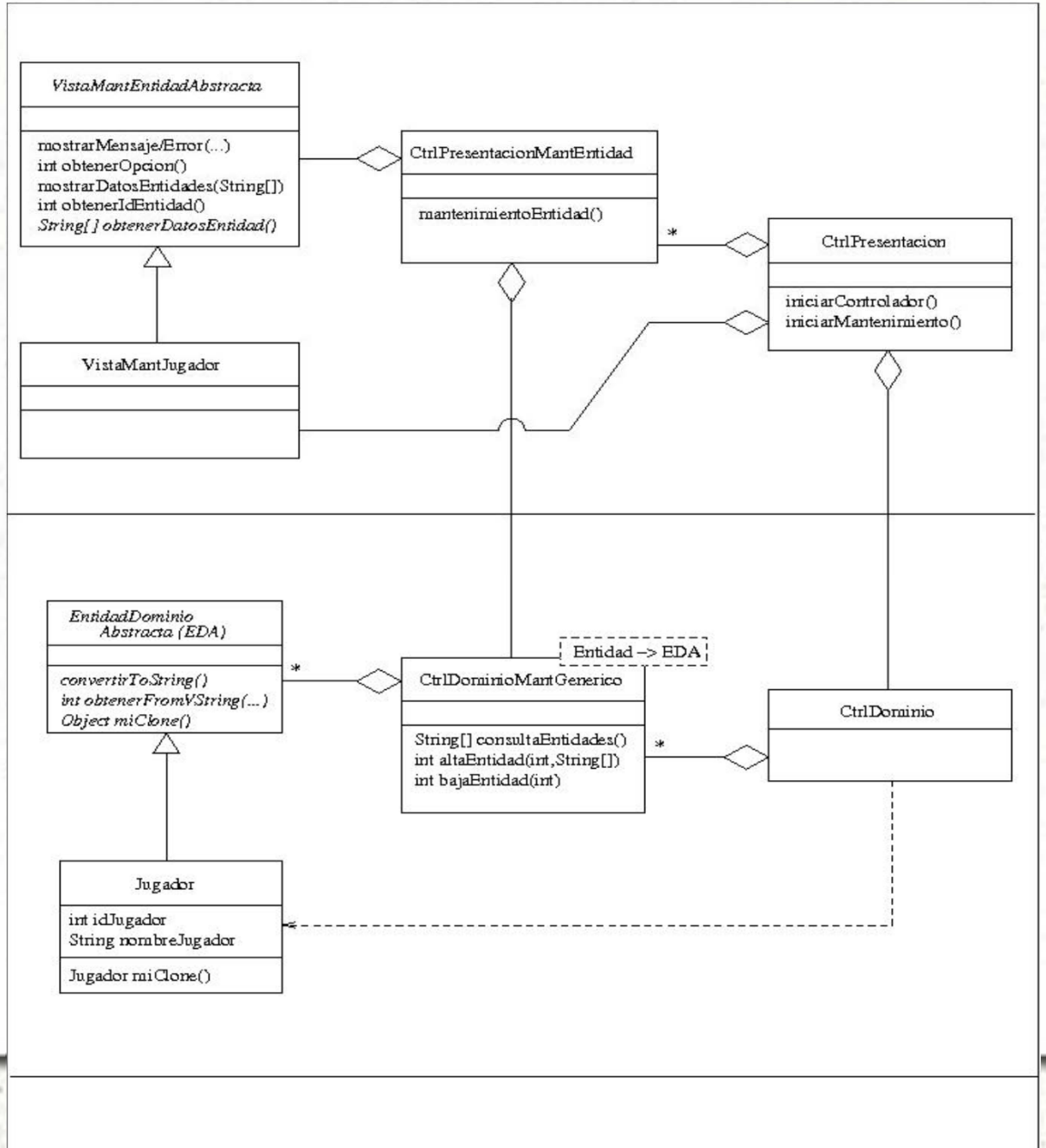


Ejemplo Arquitectura 3 capas

Cambios en la Capa de Presentación:

- VistaMantEntidadAbstracta, clase **abstracta** con la misma estructura: cambios cosméticos + un parámetro *String* con el nombre de la clase a mantener
- Sólo hay una operación que no se puede generalizar, obtenerNombreJugador -> se hace abstracta y se transforma en *Vector<String> obtenerDatosEntidad()*
- Se crea una nueva clase VistaMantJugador que hereda de VistaMantEntidadAbstracta e implementa la operación anterior

Ejemplo Arquitectura 3 capas



Ejemplo Arquitectura 3 capas

Para incluir el mantenimiento de una nueva clase (por ejemplo Equipo):

- Crear la clase Equipo, que herede de EDA e implemente:
  - Constructoras y get/setAtributo para cada atributo
  - Los métodos concretos heredados de los abstractos de EDA
- Crear una vista específica VistaMantEquipo que herede de VMEA e implemente el método *obtenerDatosEntidad()*
- Modificar CtrlDominio para incluir una nueva agregación del CtrlDominioMantGenerico<Equipo>
- Modificar CtrlPresentacion:
  - Añadir una nueva agregación para el CtrlPresentacionMantEntidad correspondiente al mantenimiento de Equipo
  - Añadir una nueva agregación para la VistaMantEquipo
  - Cambiar el método *iniciarMantenimiento()* para incluir el mantenimiento de Equipo

Ejemplo Arquitectura 3 capas

La clase CtrlDominio

```
public class CtrlDominio {

  // Controlador de Dominio para el Mantenimiento de Jugador
  private CtrlDominioMantGenerico CDmj; //*** Agregacion

  public CtrlDominio() {
    CDmj = new CtrlDominioMantGenerico<Jugador>(new Jugador());
  }

  public CtrlDominioMantGenerico getCtrlDominioMantJugador() {
    return CDmj;
  }

  // Esto es un ejemplo: En un caso real habria mas cosas...
}
```

Ejemplo Arquitectura 3 capas

La clase CtrlDominioMantGenerico

```
import java.util.*;

public class CtrlDominioMantGenerico<Entidad extends EntidadDominioAbstracta> {

  //*** Los datos se guardan en un TreeMap con clave = idEntidad (int)
  private TreeMap<Integer,Entidad> Entidades; //*** Agregacion

  //*** Variable para poder hacer las altas sin hacer un "new"
  //Entidad NewEnt = new Entidad(); //*** Error de compilacion
  Entidad NewEnt;

  public CtrlDominioMantGenerico (Entidad ent) {
    Entidades = new TreeMap<Integer,Entidad>();
    NewEnt = ent; //*** El "new" hay que hacerlo fuera
  }

  public Vector<String> consultaEntidades() {
    //*** La informacion se devuelve en estructuras generales (Vector<String>)
    Vector<String> sdatos = new Vector<String>();
    Set setkeys = Entidades.keySet();
    Iterator iterkeys = setkeys.iterator();
    while (iterkeys.hasNext()) {
      Integer ide = (Integer) iterkeys.next();
      Entidad ent = Entidades.get(ide);
      String s = ent.convertirToString();
      sdatos.add(s);
    }
    return sdatos;
  }

  private boolean existeEntidad (int ide) { //*** private
    return Entidades.containsKey(new Integer(ide));
  }
```

Ejemplo Arquitectura 3 capas

```
public int altaEntidad (int ide, Vector<String> datos) throws CloneNotSupportedException
    //*** La informacion viene en estructuras generales (Vector<String>)


    if (existeEntidad(ide))
      return 1;
    else //*** Comprobacion de consistencia (innecesaria)
    //*** El identificador esta en la posicion 0
    if (ide != (new Integer(datos.get(0))).intValue())
      return 2;
    else { // Alta del jugador
      int coderr = NewEnt.obtenerFromVectorString(ide,datos);
      if (coderr != 0) return coderr;
      Entidades.put(new Integer(ide),(Entidad)NewEnt.miClone());
    }
    return 0;
  }

  public int bajaEntidad (int ide) {

    if (!existeEntidad(ide))
      return 1;
    else { // Baja del jugador
      Entidades.remove(new Integer(ide));
    }
    return 0;
  }
}
```

Ejemplo Arquitectura 3 capas

La clase EntidadDominioAbstracta

```
import java.util.*;

public abstract class EntidadDominioAbstracta {

    abstract String convertirToString();

    abstract int obtenerFromVectorString(int ide, Vector<String> datos);

    abstract Object miClone() throws CloneNotSupportedException;

}
```

Ejemplo Arquitectura 3 capas

La clase Jugador

```
import java.util.*;

public class Jugador extends EntidadDominioAbstracta implements Cloneable {

  private int idJugador;
  private String nombreJugador;

  public Jugador() { }

  public int getId() { return idJugador; }

  public String getNombre() { return nombreJugador; }

  public boolean setId (int id) {
    idJugador = id;
    return true;
  }

  public boolean setNombre (String nombre) {
    nombreJugador = nombre;
    return true;
  }

  public String convertirToString() {
    String s = "";
    s += this.getId(); s += " ";
    s += this.getNombre();
    return s;
  }


  public int obtenerFromVectorString (int id, Vector<String> datos) {
    if (!this.setId(id)) return -1;       // Esto no pasara nunca
    //*** El nombre esta en la posicion 1
    String nombre = datos.get(1);
    if (!this.setNombre(nombre)) return -1; // Esto no pasara nunca
    return 0;
  }

  public Jugador miClone() throws CloneNotSupportedException {
    return (Jugador)this.clone();
  }
}
```

Ejemplo Arquitectura 3 capas

La clase Main

```
public class Main {
  private static CtrlPresentacion CP;
  public static void main (String[] args) throws Exception {
    // Inicializaciones generales, etc
    CP = new CtrlPresentacion();
    CP.iniciarControlador();
    // Esto es un ejemplo: Antes de llegar aqui pasara por otros sitios
    CP.iniciarMantenimiento();
  }
}
```

Ejemplo Arquitectura 3 capas

La clase CtrlPresentacion

```
public class CtrlPresentacion {

  // Controlador de Dominio
  private CtrlDominio CD;                    //*** Agregacion
  // Controlador de Dominio para el Mantenimiento de Jugador
  // (la agregacion esta en el CD / getCtrlDominioMantJugador / transitividad)
  private CtrlDominioMantGenerico CDmj;
  // Controlador de Presentacion para el Mantenimiento de Jugador
  private CtrlPresentacionMantEntidad CPmj; //*** Agregacion
  // Vista para el Mantenimiento de Jugador
  private VistaMantJugador Vmj;              //*** Agregacion

  public CtrlPresentacion() {
    CD = new CtrlDominio();
    CDmj = CD.getCtrlDominioMantJugador();
    Vmj = new VistaMantJugador("Jugador");
  }

  public void iniciarControlador() throws Exception {
    //  - ...
  }

  public void iniciarMantenimientoJugador() throws Exception {
    CPmj = new CtrlPresentacionMantEntidad ("Jugador",Vmj,CDmj);
    CPmj.mantenimientoEntidad();
  }

  public void iniciarMantenimiento() throws Exception {
    iniciarMantenimientoJugador();
  }
}
```

Ejemplo Arquitectura 3 capas

La clase CtrlPresentacionMantEntidad

```
import java.util.*;

public class CtrlPresentacionMantEntidad {

  private String nombreClase;

  // Vista para el Mantenimiento de la Entidad
  private VistaMantEntidadAbstracta Vm; //*** Agregacion
  // Controlador de Dominio para el Mantenimiento de Entidad
  private CtrlDominioMantGenerico CDm;  //*** Agregacion

  CtrlPresentacionMantEntidad (String nom, VistaMantEntidadAbstracta v,
  CtrlDominioMantGenerico c)
    nombreClase = nom;
    Vm = v;  //*** Asigna la vista de mantenimiento
    CDm = c; //*** Asigna el controlador de dominio de mantenimiento
  }

  public void mantenimientoEntidad() throws Exception {
    int opcion = -1;
    while (opcion != 0) {
      //*** La vista solo recoge y/o muestra datos
      opcion = Vm.obtenerOpcion();
      switch (opcion) {
        case 0: break;
        case 1: consultaEntidades(); break;
        case 2: altaEntidad(); break;
        case 3: bajaEntidad(); break;
        default: break;
      }
    }
  }

  private void consultaEntidades() throws Exception { //*** private
    Vm.mostrarMensaje('-',"Consulta datos " + nombreClase);
    //*** El que realmente hace el trabajo es el controlador de dominio
    Vector<String> datos = CDm.consultaEntidades();
    //*** La vista solo recoge y/o muestra datos
    Vm.mostrarDatosEntidades(datos);
  }
```

Ejemplo Arquitectura 3 capas

```
  private void altaEntidad() throws Exception { //*** private
    Vm.mostrarMensaje('-',"Alta de " + nombreClase);
    int id = Vm.obtenerIdEntidad();
    Vector<String> datosEntidad = Vm.obtenerDatosEntidad();
    //*** El que realmente hace el trabajo es el controlador de dominio
    Vector<String> datosAlta = new Vector<String>();
    datosAlta.add((new Integer(id)).toString());
    int n = datosEntidad.size();
    for (int i=0; i<n; ++i)
      datosAlta.add(datosEntidad.get(i));
    int codierr = CDm.altaEntidad(id,datosAlta);
    //*** La vista solo recoge y/o muestra datos
    switch (codierr) {
      case 0:  Vm.mostrarMensaje('-',"Alta efectuada"); break;
      case 1:  Vm.mostrarError(nombreClase + " ya existe"); break;
      case 2:  Vm.mostrarError("Error de inconsistencia"); break;
      default: Vm.mostrarError("Error imposible "+codierr); break;
    }
  }

  private void bajaEntidad() throws Exception { //*** private
    Vm.mostrarMensaje('-',"Baja de " + nombreClase);
    int id = Vm.obtenerIdEntidad();
    //*** El que realmente hace el trabajo es el controlador de dominio
    int codierr = CDm.bajaEntidad(id);
    //*** La vista solo recoge y/o muestra datos
    switch (codierr) {
      case 0:  Vm.mostrarMensaje('-',"Baja efectuada"); break;
      case 1:  Vm.mostrarError(nombreClase + " no existe"); break;
      default: Vm.mostrarError("Error imposible "+codierr); break;
    }
  }
}
```

Ejemplo Arquitectura 3 capas

La clase VistaMantEntidadAbstracta

```
import java.util.*;

public abstract class VistaMantEntidadAbstracta {

  protected String nombreClase;

  protected inout io = new inout();
  private int nOpciones = 0;


  public VistaMantEntidadAbstracta (String nom) {
    nombreClase = nom;
  }

  ////////////////////////////////////////////////
  // Funciones de visualizacion y opciones del menu
  ////////////////////////////////////////////////

  private void mostrarVista() throws Exception {
    io.writeln("");
    String mensaje = "Menu Mantenimiento " + nombreClase;
    mostrarMensaje('*',mensaje);
    io.writeln("0 - Salir");
    io.writeln("1 - Consulta datos " + nombreClase); ++nOpciones;
    io.writeln("2 - Alta " + nombreClase); ++nOpciones;
    io.writeln("3 - Baja " + nombreClase); ++nOpciones;
    io.write("Opcion: ");
  }

  public void mostrarMensaje(char c, String mensaje) throws Exception {
    int n = mensaje.length();
    for (int i=0; i<n; ++i) io.write(c); io.writeln("");
    io.writeln(mensaje);
    for (int i=0; i<n; ++i) io.write(c); io.writeln("");
  }

  public void mostrarError(String mensaje) throws Exception {
    io.writeln(""); io.writeln("ERROR: "+mensaje); io.writeln("");
  }
```

Ejemplo Arquitectura 3 capas

```
public int obtenerOpcion() throws Exception {
  int opcion = -1;
  while (opcion < 0 || opcion > nOpciones) {
    mostrarVista();
    opcion = io.readint();
  }
  io.writeln("");
  return opcion;
}

////////////////////////////////////////////////////////
// Funciones de visualizacion de datos de las entidades
////////////////////////////////////////////////////////

public void mostrarDatosEntidades (Vector<String> datos) throws Exception {
  int n = datos.size();
  for (int i=0; i<n; ++i)
    io.writeln(nombreClase + ": " + datos.get(i));
}

////////////////////////////////////////////////
// Funciones de obtencion de datos de la entidad
////////////////////////////////////////////////

public int obtenerIdEntidad() throws Exception {
  io.write("Id de " + nombreClase + ": ");
  int id = io.readint();  String r = io.readline();
  return id;
}

// Funcion abstracta para la obtencion del resto de datos
public abstract Vector<String> obtenerDatosEntidad() throws Exception;

}
```

Ejemplo Arquitectura 3 capas

La clase VistaMantJugador

```
import java.util.*;

public class VistaMantJugador extends VistaMantEntidadAbstracta {

  public VistaMantJugador (String nom) {
    super(nom);
  }

  public Vector<String> obtenerDatosEntidad() throws Exception {
    Vector<String> vs = new Vector<String>();

    // Nombre
    io.write("Nombre de " + nombreClase + ": ");
    String nombre = io.readline();
    vs.add(nombre);

    return vs;
  }
}
```