Control 1 Prueba 1 Paralelo 2

Competencia a evaluar: El estudiante debe ser capaz de construir el modelo del dominio, contratos, diagrama de clases y código Java asociado, siguiendo arquitectura propuesta

Problema

Se debe construir una aplicación que lea y procese un archivo. Cada línea del archivo contiene un RUT y un código de departamento, representando que la persona con ese RUT es el dueño del departamento indicado. Considere que un dueño puede tener hasta 7 departamentos y que un departamento tiene un solo dueño. Después de procesar todo el archivo, la aplicación debe realizar lo siguiente:

1. Desplegar por pantalla los datos de los departamentos (códigos) que posee un determinado dueño, cuyo RUT se ingresa desde el teclado
2. Desplegar por pantalla el dueño (RUT) de un departamento, cuyo código se ingresa desde el teclado.

Se pide (escrito en papel):

a. Modelo del dominio completo
b. Contratos de la interfaz del "sistema"
c. Diagrama de clases completo
d. Código Java de:
  i. Clase `Departamento`
  ii. Clase `ListaDepartamentos`
  iii. Código que implementa la interfaz del "sistema"
  iv. Clase App completa (la que contiene el método `main`)

Notas

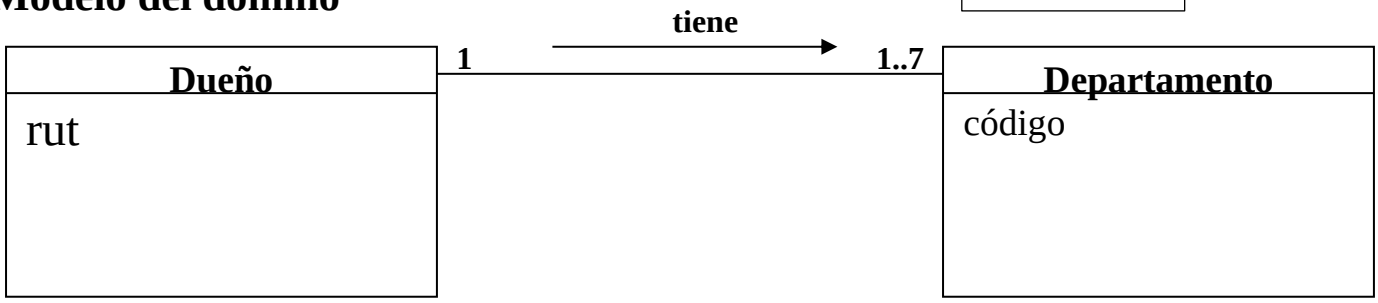
- Trabaje con excepciones para las precondiciones
- No es necesario hacer los `import`
- No es necesario trabajar con paquetes
- Para el caso de atributos de tipo primitivo, tanto en el diagrama de clases como en el código solo escriba `get`… y `set`…
- Si usa `toString()`, escriba el código

Recopile lo que escribió en los puntos anteriores, genere un solo PDF y suba dicho archivo PDF al trabajo en CampusVirtual.

Solución control 1

a. Modelo del domino

0.2 puntos



b. Contratos

1 punto

Operación	**Ingresar departamento (**codigo**)**
Descripción	Se ingresa el departamento a la lista general de departamentos
Precondiciones	
Postcondiciones	Departamento ingresado a la lista general de departamentos

Operación	**Ingresar dueño (**rut**)**
Descripción	Se ingresa el dueño
Precondiciones	
Postcondiciones	Dueño ingresado a la lista general de dueños

Operación	**Asociar dueño departamento (**código departamento, rut**)**
Descripción	Se ingresa el departamento a la lista de departamentos del dueño Se asocia el departamento con el dueño
Precondiciones	Que exista el dueño Que exista el departamento
Postcondiciones	• Departamento ingresado a la lista de departamentos del dueño • Departamento asociado con el dueño

Operación	**Obtener datos de los departamentos de un dueño (**rut**)**
Descripción	Se obtiene el código de cada uno de sus departamentos de los cuales es dueño
Precondiciones	Que exista el dueño
Post condiciones	

Operación	**Obtener datos del dueño de un departamento (**codigo**)**
Descripción	Se obtienen los datos del dueño del departamento
Precondiciones	Que exista el departamento
Post condiciones	

1 punto: 0.2 cada contrato

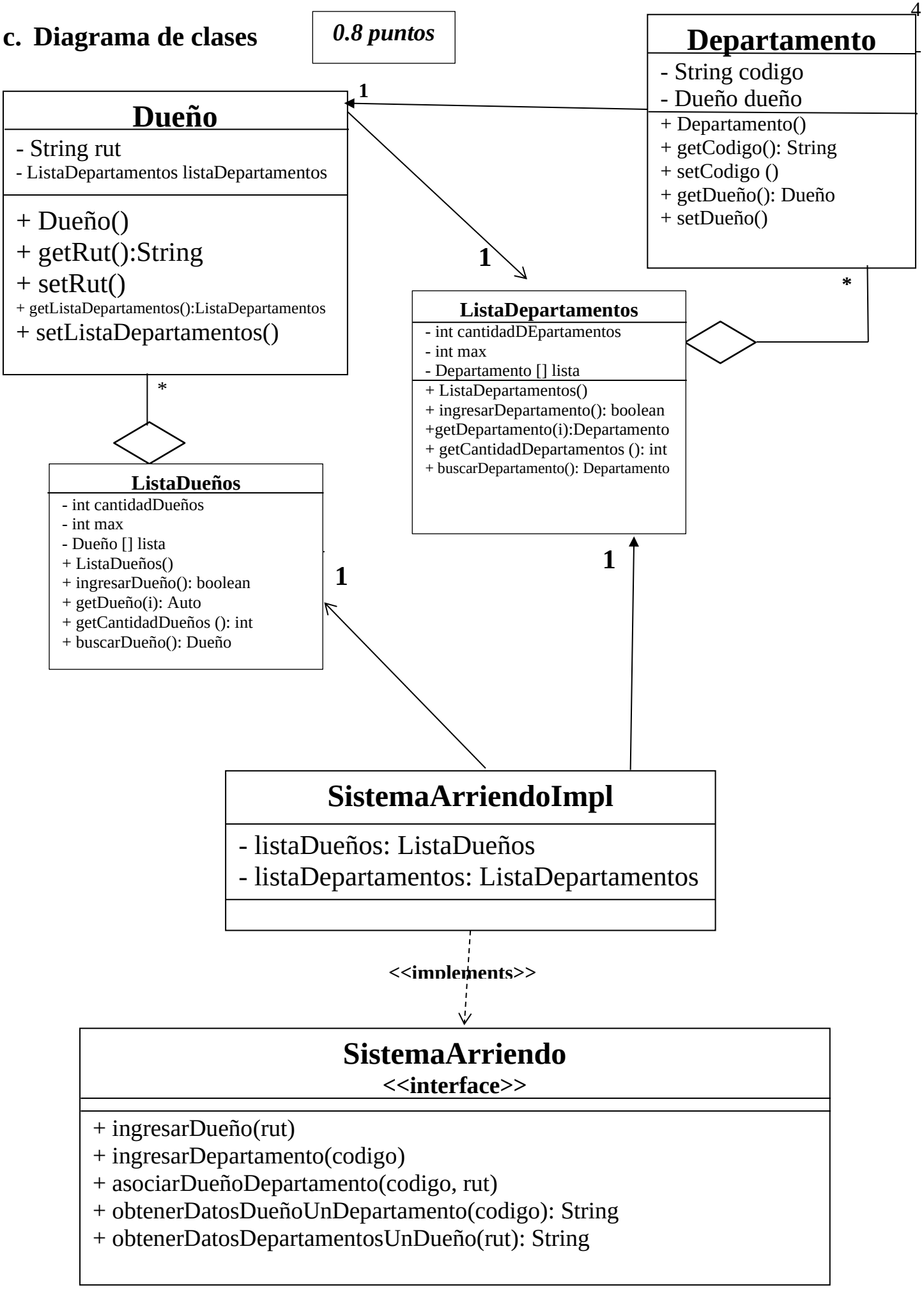
Observación

En vez de los 3 primeros contratos, se podrían tener 2 contratos:

- *Ingresar departamento (código)*
- *Ingresar dueño (código, rut): Sería el equivalente a ingresarDueño y asociarDueñoDepartamento*

c. Diagrama de clases

0.8 puntos



d. Código Java

4 puntos

0.5 puntos

```java
package cl.ucn.ei.pa.sistemaCompraventa.dominio;

public class Departamento {
	private String codigo;
	private Dueño dueño;

	public Departamento(String codigo) {
		this.codigo = codigo;
		this.dueño = null;
	}

	public String getCodigo() {
		return codigo;
	}

	public void setCodigo(String codigo) {
		this.codigo = codigo;
	}

	public Dueño getDueño() {
		return dueño;
	}

	public void setDueño(Dueño dueño) {
		this.dueño = dueño;
	}

}
```

```java
package cl.ucn.ei.pa.sistemaCompraventa.dominio;

import cl.ucn.ei.pa.sistemaCompraventa.logica.ListaDepartamentos;

public class Dueño {
	private String rut;
	private ListaDepartamentos listaDepartamentos;

	public Dueño(String rut) {
		this.rut = rut;
		listaDepartamentos = new ListaDepartamentos(7);
	}

	public String getRut() {
		return rut;
	}

	public void setRut(String rut) {
		this.rut = rut;
	}

	public ListaDepartamentos getListaDepartamentos() {
		return listaDepartamentos;
	}

	public void setListaDepartamentos(ListaDepartamentos listaDepartamentos) {
		this.listaDepartamentos = listaDepartamentos;
	}

}
```

No se pide

No se pide

```java
package cl.ucn.ei.pa.sistemaCompraventa.logica;

import cl.ucn.ei.pa.sistemaCompraventa.dominio.Departamento;

public class ListaDepartamentos {
	private int max;
	private int cantidadDepartamentos;
	private Departamento [] lista;

	public ListaDepartamentos (int max) {
		lista = new Departamento[max];
		cantidadDepartamentos = 0;
		this.max = max;
	}

	public boolean ingresarDepartamento(Departamento departamento) {
		if (cantidadDepartamentos < max) {
			lista[cantidadDepartamentos]= departamento;
			cantidadDepartamentos++;
			return true;
		}
		else {
			return false;
		}
	}

	public Departamento getDepartamento (int i) {
		if (i < cantidadDepartamentos) {
			return lista[i];
		}
		else {
			return null;
		}
	}

	public Departamento buscarDepartamento(String codigo) {
		int i = 0;
		while(i<cantidadDepartamentos && !lista[i].getCodigo().equals(codigo)){
			i++;
		}
		if (i == cantidadDepartamentos) {
			return null;
		}
		else {
			return lista[i];
		}
	}

	public int getCantidadDepartamentos() {
		return cantidadDepartamentos;
	}

}
```

0.5 puntos

```java
package cl.ucn.ei.pa.sistemaCompraventa.logica;

import cl.ucn.ei.pa.sistemaCompraventa.dominio.Dueño;


public class ListaDueños {
	private int max;
	private int cantidadDueños;
	private Dueño [] lista;

	public ListaDueños (int max) {
		lista = new Dueño[max];
		cantidadDueños = 0;
		this.max = max;
	}

	public boolean ingresarDueño(Dueño Dueño) {
		if (cantidadDueños < max) {
			lista[cantidadDueños]= Dueño;
			cantidadDueños++;
			return true;
		}
		else {
			return false;
		}
	}

	public Dueño buscarDueño(String rut) {
		int i = 0;
		while(i < cantidadDueños && !lista[i].getRut().equals(rut)) {
			i++;
		}
		if (i == cantidadDueños) {
			return null;
		}
		else {
			return lista[i];
		}
	}

	public Dueño getDueño (int i) {
		if (i < cantidadDueños) {
			return lista[i];
		}
		else {
			return null;
		}
	}
}
```

No se pide

```java
package cl.ucn.ei.pa.sistemaCompraventa.logica;

public interface SistemaVentaDepartamentos {
	public boolean ingresarDepartamento(String codigo);
	public boolean ingresarDueño(String rut);
	public void asociarDueñoDepartamento(String rut, String codigo);
	public String obtenerDatosDepartamentosUnDueño(String rut);
	public String obtenerDueñoUnDepartamento(String codigo);
}
```

2 puntos

```java
package cl.ucn.ei.pa.sistemaCompraventa.logica;

import cl.ucn.ei.pa.sistemaCompraventa.dominio.Departamento;
import cl.ucn.ei.pa.sistemaCompraventa.dominio.Dueño;

public class SistemaVentaDepartamentosImpl implements SistemaVentaDepartamentos{
	private ListaDueños listaDueños;
	private ListaDepartamentos listaDepartamentos;

	public SistemaVentaDepartamentosImpl() {
		listaDueños = new ListaDueños(5);
		listaDepartamentos = new ListaDepartamentos(50);
	}

	public boolean ingresarDepartamento(String codigo) {
		Departamento Departamento = new Departamento(codigo);
		return listaDepartamentos.ingresarDepartamento(Departamento);
	}

	public boolean ingresarDueño(String rut) {
		Dueño dueño = listaDueños.buscarDueño(rut);
		if (dueño == null) {
			dueño = new Dueño(rut);
			return listaDueños.ingresarDueño(dueño);
		}
		else {
			return true;
		}
	}

	public void asociarDueñoDepartamento(String rut, String codigo) {
		Dueño dueño = listaDueños.buscarDueño(rut);
		Departamento departamento=listaDepartamentos.buscarDepartamento(codigo);
		if(dueño != null && Departamento != null) {
			Departamento.setDueño(dueño);
			dueño.getListaDepartamentos().ingresarDepartamento(Departamento);
		}
		else {
			throw new NullPointerException("No existe departamento y/o dueño");
		}
	}
```

```
    public String obtenerDatosDepartamentosUnDueño(String rut) {
        String salida ="Departamentos de un dueño\n";
        Dueño dueño = listaDueños.buscarDueño(rut);
        if(dueño != null) {
            for(int i=0;i<dueño.getListaDepartamentos().getCantidadDepartamentos();i++){
                Departamento departamento = dueño.getListaDepartamentos().getDepartamento(i);
                salida = salida+"codigo "+Departamento.getCodigo()+ "\n";
            }
            return salida;
        }
        else {
            throw new NullPointerException ("No existe el dueño");
        }
    }



    public String obtenerDueñoUnDepartamento(String codigo) {
        String salida ="Dueño del Departamento ";
        Departamento departamento=listaDepartamentos.buscarDepartamento(codigo);
        if(departamento != null) {
            salida = salida + departamento.getDueño().getRut();
            return salida;
        }
        else {
            throw new NullPointerException ("No existe el departamento");
        }
    }
}
```

1 punto

```
package cl.ucn.ei.pa.sistemaCompraventa.logica;

import java.io.IOException;

import ucn.ArchivoEntrada;
import ucn.Registro;
import ucn.StdIn;
import ucn.StdOut;

public class App {

	public static void leerDepartamentos(SistemaVentaDepartamentos sistema)
                                                                    throws IOException{
		ArchivoEntrada archivo = new ArchivoEntrada("Departamentos.txt");
		boolean ingreso = true;
		while(!archivo.isEndFile() && ingreso) {
			Registro registro = archivo.getRegistro();
			String codigo = registro.getString();
			String rut = registro.getString();
			ingreso = sistema.ingresarDepartamento(codigo);
			ingreso = sistema.ingresarDueño(rut);
			try {
				sistema.asociarDueñoDepartamento(rut,codigo);
			}catch(NullPointerException ex) {
				StdOut.println(ex.getMessage());
			}
		}
	}



	public static void main(String[] args) throws IOException{
		SistemaVentaDepartamentos sistema=new SistemaVentaDepartamentosImpl();
		leerDepartamentos(sistema);
		StdOut.print("Rut dueño: ");
		String rut = StdIn.readString();
		try {
			StdOut.println(sistema.obtenerDatosDepartamentosUnDueño(rut));
		}catch(NullPointerException ex) {
			StdOut.println(ex.getMessage());
		}

		StdOut.print("codigo: ");
		String codigo = StdIn.readString();
		try {
			StdOut.println(sistema.obtenerDueñoUnDepartamento(codigo));
		}catch(NullPointerException ex) {
			StdOut.println(ex.getMessage());
		}
	}
}
```