UNIVERSIDAD ICESI	Primer Parcial	Algoritmos y Lenguajes II

Reglas del juego

- Tal como se le dijo antes, durante este examen usted no puede pedir ABSOLUTAMENTE nada prestado a sus compañeros, ni hablar con ellos.
- Su nombre y su firma a la aceptación del compromiso de no hacer fraude, en la hoja de respuestas a este examen, deben ir en lapicero. Si emplea más de una hoja márquelas TODAS de igual forma.
- Por ningún motivo puede salir del salón, antes de terminar el examen. De manera que si se retira se considerará que terminó su trabajo.
- Puede consultar sus notas, el libro o apuntes; pero se le recomienda que lo haga cuando esté seguro de qué quiere buscar, en caso contrario estas consultas se convierten en una distracción que le hace perder tiempo.
- No se resolverá ninguna duda durante el examen, así que si algo no le queda claro y usted requiere hacer algún tipo de supuesto, ESCRÍBALO y de una breve explicación de la razón por la cual lo supone. Tenga presente que un supuesto no puede contradecir lo que se le está diciendo.
- Debe escribir con letra clara.

La forma de evaluar este parcial es la siguiente:

Ítem	Valor
Primer problema:	**2.5**
Identificar donde deben manejarse Excepciones	0.5
Implementar el manejo de Excepciones (código)	1.5
Argumentación de validaciones requeridas para evitar Excepciones no comprobadas	0.5
Segundo problema:	**2.5**
Diagrama de clases	1.5
Clases, atributos y métodos	0.5
Relaciones	0.5
Interfaces	0.5
Descripción de métodos	1.0

Punto 1

Un objeto de tipo BufferedReader permite leer caracteres desde un archivo de datos, de manera similar a como pueda hacerse con un objeto de tipo Scanner.

El siguiente código permite leer la información de un archivo de texto llamado “archivo.txt”, línea por línea a través del método +readLine():String de BufferedReader. A medida que se va leyendo cada línea se obtiene una subcadena de la misma a partir de la posición que indique el usuario. Al final, el sistema mostrará todas las subcadenas obtenidas.

A continuación se presenta el código con su correspondiente análisis:

```
package primerparcial;
import java.io.*;
import java.util.*;

public class Enunciado {

/* Método: main
 * Proceso: Crea un BufferedReader asociado al archivo especificado, llama al método
 * +leerDesdeArchivo(BufferedReader):ArrayList para obtener las líneas del archivo recuperando las
 * subcadenas que se obtuvieron a partir de las posiciones que el usuario indicó.  Estas
 * subcadenas son retornadas por el método +leerDesdeArchivo(BufferedReader):ArrayList en un
 * ArrayList.  Una vez se obtienen las subcadenas, se invoca al método que se encargará de
```

```
 * mostrarlas por pantalla.
 * */
	public static void main(String[] args) {
		BufferedReader elBuffer= new BufferedReader(new FileReader("d:\\archivo.txt"));
		ArrayList lista=leerDesdeArchivo(elBuffer);
		mostrarSubcadenas(lista);
	}
/* Método: leerDesdeArchivo
 * Parámetros: El objeto BufferedReader que permitirá leer cada línea del archivo.
 * Retorno: Un ArrayList con las subcadenas.
 * Proceso: Lee cada línea mientras no se haya terminado el archivo, y a medida que va leyendo
 * cada una, pregunta al usuario desde que posición desea obtener la subcadena de la línea actual.
 * Obtiene la subcadena a partir de la posición indicada y la almacena en un ArrayList.
 * */
public static ArrayList leerDesdeArchivo(BufferedReader lectorDesdeArchivo)
{
  Scanner lector=new Scanner(System.in);
  boolean terminado=false;
  int posición;
  ArrayList subcadenas=new ArrayList();
  while(!terminado)
  {
    String linea=lectorDesdeArchivo.readLine();
    if(linea==null)
    {
       terminado=true;
    }
    else
    {
       System.out.println("Digite la posición a partir de la cual desea obtener la subcadena");
       posición=lector.nextInt();
       String subCadena=linea.substring(posición);
       subcadenas.add(subCadena);
    }
   }
 return subcadenas;
}

/*Método: mostrarSubcadenas
 *Parámetros: El ArrayList con las subcadenas
 *Salidas: Cada una de las subcadenas
 *Proceso: Recorre el ArrayList y va mostrando la subcadena almacenada en cada posición.
 * */
	public static void mostrarSubcadenas(ArrayList subCadenas)
	{
		for(int cont=0;cont<subCadenas.size();cont++)
		{
			System.out.println((String)subCadenas.get(cont));
		}
	}
}
```

Lo que usted debe hacer es lo siguiente:

- Implementar el manejo de excepciones en los métodos en los que usted considere necesario. Deberá reescribir el o los métodos mostrando claramente como haría el manejo de la excepción o de las excepciones.
- Si considera que debería haberse hecho validaciones para evitar que se presenten excepciones no comprobadas, indique en que parte del código debería hacerse y explique con sus palabras cómo lo haría.

Tenga en cuenta la siguiente documentación del API:

Documentación de constructores utilizados

FileReader: Permite instanciar un flujo al archivo especificado en la ruta que se define como parámetro.

`public` **`FileReader`**`(String fileName)`

Creates a new `FileReader`, given the name of the file to read from.

Parameters:
`fileName` - the name of the file to read from
Throws:
`FileNotFoundException` - if the named file does not exist, is a directory rather than a regular file, or for some other reason cannot be opened for reading.

BufferedReader: Permite instanciar un objeto de tipo BufferedReader que permite la lectura de los caracteres desde el flujo asociado al archivo.

`public BufferedReader(Reader in)`

Creates a buffering character-input stream that uses a default-sized input buffer.
Parameters:
`in` - A Reader

Documentación de métodos utilizados

readLine – Método de la clase BufferedReader

`public String readLine()`

Reads a line of text. A line is considered to be terminated by any one of a line feed ('\n'), a carriage return ('\r'), or a carriage return followed immediately by a linefeed.
Returns:
A String containing the contents of the line, not including any line-termination characters, or null if the end of the stream has been reached
Throws:
`IOException` - If an I/O error occurs

Substring – Método de la clase String

`public String substring(int beginIndex)`

Returns a new string that is a substring of this string. The substring begins with the character at the specified index and extends to the end of this string.
Parameters:
`beginIndex` - the beginning index, inclusive.
Returns:
the specified substring.
Throws:
`IndexOutOfBoundsException` - if `beginIndex` is negative or larger than the length of this `String` object.

Documentación de excepciones

java.lang
Class IndexOutOfBoundsException

`java.lang.Object`
  └ `java.lang.Throwable`
    └ `java.lang.Exception`
      └ `java.lang.RuntimeException`
        └ **`java.lang.IndexOutOfBoundsException`**

java.io
Class IOException

`java.lang.Object`
  └ `java.lang.Throwable`
    └ `java.lang.Exception`
      └ **`java.io.IOException`**

java.io
Class FileNotFoundException

java.lang.Object
  └ java.lang.Throwable
      └ java.lang.Exception
          └ java.io.IOException
              └ **java.io.FileNotFoundException**

Punto dos

Un parque de diversiones está necesitando el desarrollo de una aplicación orientada a objetos en Java que le permita gestionar el pago de las obligaciones que conlleva su funcionamiento. Los pagos que el parque de diversiones debe realizar corresponden a pagos de salarios a empleados, pagos de servicios a proveedores (otras empresas que prestan servicios, como por ejemplo el mantenimiento a las atracciones, las empresas de servicios públicos, etc), y pagos de dividendos a los propietarios del parque.

De los empleados, se almacena el número del documento de identificación, su nombre y el salario básico que ganan en el mes.

Es importante tener en cuenta que existen dos tipos de empleados, los operarios y los empleados administrativos.

En el caso de los empleados que trabajan como operarios del parque debe almacenarse el número de horas extras trabajadas. En el caso de los empleados administrativos, se almacena además el valor del auxilio de transporte al que tiene derecho el empleado.

El pago mensual que se le hace a un empleado depende de si es operario o administrativo. Los operarios reciben un pago mensual se constituye del salario base más $12.000 por cada hora extra laborada. En el caso de los empleados administrativos el salario está constituido por el salario base menos una retención del 10% más el valor del auxilio de transporte.

En el caso de los proveedores, o empresas que prestan servicios al parque, se quiere registrar en el sistema el número de identificación, la razón social, una descripción del servicio que prestan y el costo mensual de su servicio. El valor a pagar durante el mes a cada proveedor corresponde al costo mensual del servicio más el 16% por concepto de IVA.

Finalmente, el pago de dividendos a los propietarios del parque depende del porcentaje de propiedad sobre el mismo. Por ejemplo, si un propietario tiene un 50% de participación, el valor que debe pagársele corresponde al 50% de las utilidades del parque en el mes.

En el caso de los propietarios se almacena en el sistema, además del porcentaje de participación, su número de identificación y nombre.

Usted debe:

1. Realizar el diagrama de clases completo para que el sistema pueda registrar la información y desplegar las consultas solicitadas.
2. Realizar la descripción de TODOS los métodos necesarios para consultar el valor a pagar a un empleado, un proveedor o un propietario, dado su número de identificación.
3. Realizar la descripción de los métodos necesarios para mostrar el listado de propietarios ordenados de forma ASCENDENTE según su número de identificación.