Programación 3
MMMMMM 2023, Ciclo IV, Unidad I, xv + 123.
Universidad de El Salvador

Semana 2 – Grupo Teorico

La Poosexualidad
Escuela de Ingeniería en Sistemas

Comunismo
Universidad de El Salvador

Abstract

This guide of the Austrian Journal of Statistics (AJS) shows the required style of manuscripts submitted to the Austrian Journal of Statistics. We strongly recommend to follow the given instructions exactly already when writing the paper. When submitting papers to AJS, it is essential to follow these guidelines. Manuscripts submitted without consideration of the AJS style guide will most likely not be forwarded for review, but rejected with the possibility of resubmission.

Keywords: keywords, not capitalized, Java.

1. Clase Martes

Comentario 1 *Esta semana iniciaron los laboratorios*

Implementación de Estructuras de Datos → vectores de objetos

- Bases de datos

1.1. Estructura de Datos clásicas

Son las que manejamos en memoria.

$$\text{Arreglos} \begin{cases} \texttt{int} \\ \texttt{float} \\ \texttt{String} \end{cases}$$

Cuando tenemos asociaciones en con multiplicidad mayor a 1.

Listing 1: En un diagrama de UML

```
ruedas: ArrayList <Ruedas>
```

`ArrayList` es una clase y son los arreglos más fáciles de utilizar en Java. No necesitan que se defina la cantidad.

Algunos métodos de `ArrayList` son:

- add
- get
- remove

Comentario 2 *Cuando no hay multiplicidad → 1.*

Comentario 3 *A cada atributo hay que hacerle un `set` y `get`*

1.2. Diagramas de UML

Para los ArrayList no es el mismo `set` y `get` normales. Como es un vector debemos ser capaces de irle agregando.

En el `set` y `get` se requeriría de toda la lista.

Listing 2: Método agregar

```
agregarRuedas(objRueda){
  ruedas.add(objRueda);
}
```

Listing 3: Método set

```
setRuedas(objRueda, n){
  ruedas.set(objRueda, n);
}
```

Listing 4: Método get

```
getRuedas(n){
  return ruedas.get(n);
}
```

Listing 5: Método vacio

```
ruedasVacia(){
  return ruedas.isEmpty();
}
```

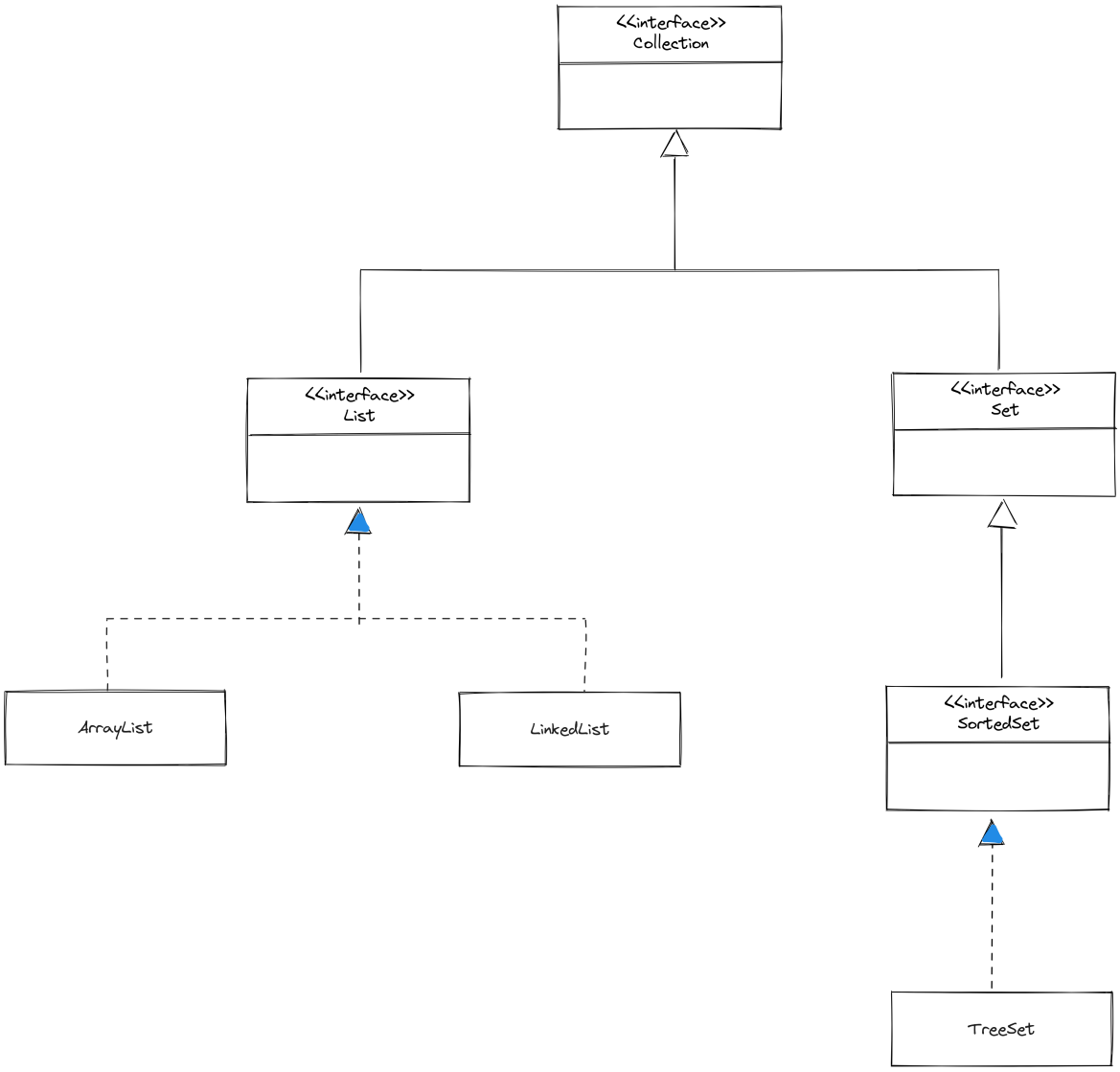
Comentario 4 *Falta el método de remove.*

1.3. API

Generalmente se importa todo el java.util

Listing 6: Agregar paquete

```
java.util.*
```

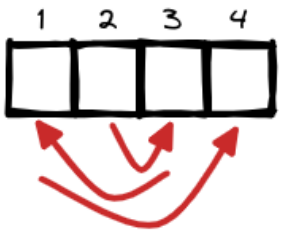


El Set no permite duplicados.

- Listas: los objetos se manejan como vectores

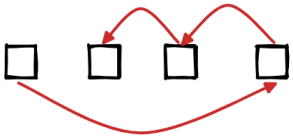
Comentario 5 *No explica la diferencia por que no es clase de ESD115.*

Las listas enlazadas usan punteros e índices.



Comentario 6 *En resumen: usar los* `ArrayList`

En el TreeSet no hay índices, trabaja con posiciones en la memoria RAM.



Comentario 7 *Las clases agregan código a los métodos de una interfaz y le agrega sus métodos.*

Se puede realizar lo siguiente:

Listing 7: Método agregar

```
main(){
  Collection lista = new ArrayList <Estudiante>();
  List lista2 = new ArrayList <Estudiante>();
  ArrayList lista3 = new ArrayList <Estudiante>();

  lista2 = (List)lista; //Esto es porque lista es Collection
  lista = lista2;
  lista3 = (ArrayList)lista;
}
```

1.4. Ejemplos

Ejemplos en Apache Netbeans

Listing 8: Código del `for`

```
for(String cadena : listaCadenas)
//el segundo argumento debe ser del mismo tipo que el primero.
```

Comentario 8 *Le dio error en el código -_-*

2. Clase Viernes

2.1. Implementación de los diagramas de clase

Recordar que el método `main` es indispensable.

Comentario 9 *Repaso de PRN2*

Las clases solo almacenan datos → lecturas e impresiones se dan en el `main`.
Las variables guardan la dirección de memoria.

Comentario 10 *Recordar que significaban en código las asociaciones y dependencias de un diagrama de UML.*

Comentario 11 *En la PE no van a correr el programa.*

Constructores

Listing 9: Constructor

```
crear() //Constructor sin parametros
```

Dentro de un constructor con parametros utilizar los `set` y `get`.

Listing 10: Dentro de un constructor

```
setAtributo(atributo); //Valido

this.atributo=atributo; // No hay validacion del dato
```

Clase Object

Comentario 12 *Implícitamente todas las clases heredan del Object.*

Métodos de la clase object son:

- `equals`
- `toString`

Affiliation: