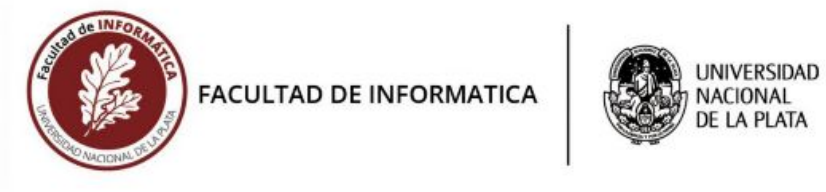
Orientación a Objetos I

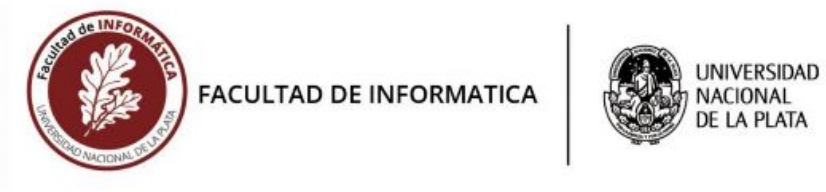
Agenda de la semana
del 28 de octubre



Temario

Ejercicios de la semana

- Ejercicio 21. Bag
- Ejercicio 22. Estadísticas del Cliente de Correo
- Ejercicio 23. Mercado de objetos



Bag - Primera parte

Ejercicio 21. Bag

Primera parte

Un Map en Java es una estructura de datos que asocia objetos que actúan cómo claves (*keys*), a otros objetos que actúan cómo valores (*values*). En otros lenguajes también se llaman Dictionary (Diccionario). Cada clave está vinculada a un único valor; no pueden existir claves duplicadas en un mapa, aunque sí podrían haber valores repetidos. A los pares clave-valor se los denomina entradas (*entries*).

Para definir un Map, es necesario indicar el tipo que tendrán sus claves y valores: por ejemplo, una variable de tipo `Map<String, Integer>` define un mapa en donde sus claves son de tipo *String*, y sus valores de tipo *Integer*. Observe que `Map<K, V>` es una interfaz.

FACULTAD DE INFORMATICA

UNIVERSIDAD NACIONAL DE LA PLATA

Bag - Primera parte

a. Lea la documentación de Map en Java y responda:
    a. ¿Qué implementaciones provee Java para utilizar un Map? ¿Cuáles de ellas son destinadas a uso general?
    b. Investigue cómo consultar sí un mapa contiene una determinada clave (key). Explique qué métodos deben implementar las claves para que esto funcione correctamente
    c. ¿Con qué método se puede recuperar el objeto asociado a una clave? ¿Qué pasa sí la clave no existe en el mapa?
    d. Investigue cómo agregar claves y valores a un mapa. ¿Qué pasa sí la clave ya se encontraba en el mapa? ¿Permite agregar claves y/o valores nulos?
    e. Determine cómo se pueden eliminar claves y valores de un mapa. ¿Es necesario controlar la presencia de alguno de ellos?
    f. Investigue cómo reemplazar un valor en un mapa
    g. Teniendo en cuenta los métodos `keys()`, `values()` y `entrySet()`, explique de qué formas se puede iterar un mapa ¿Es posible utilizar streams?

V	put(K key, V value)	Associates the specified value with the specified key in this map (optional operation).
void	putAll(Map<? extends K,? extends V> m)	Copies all of the mappings from the specified map to this map (optional operation).
default V	putIfAbsent(K key, V value)	If the specified key is not already associated with a value (or is mapped to null) associates it with the given value and returns null, else returns the current value.
V	remove(Object key)	Removes the mapping for a key from this map if it is present (optional operation).
default boolean	remove(Object key, Object value)	Removes the entry for the specified key only if it is currently mapped to the specified value.
default V	replace(K key, V value)	Replaces the entry for the specified key only if it is currently mapped to some value.

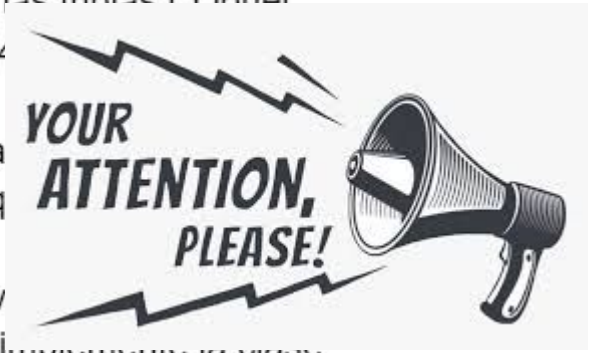
boolean	containsKey(Object key)	Returns true if this map contains a mapping for the specified key.
boolean	containsValue(Object value)	Returns true if this map maps one or more keys to the specified value.
static <K,V> Map<K,V>	copyOf(Map<? extends K,? extends V> map)	Returns an unmodifiable Map containing the entries of the given Map.
static <K,V> Map.Entry<K,V>	entry(K k, V v)	Returns an unmodifiable Map.Entry containing the given key and value.
Set<Map.Entry<K, V>>	entrySet()	Returns a Set view of the mappings contained in this map.
boolean	equals(Object o)	Compares the specified object with this map for equality.
default void	forEach(BiConsumer<? super K,? super V> action)	Performs the given action for each entry in this map until all entries have been processed or the action throws an exception.
V	get(Object key)	Returns the value to which the specified key is mapped, or null if this map contains no mapping for the key.
default V	getOrDefault(Object key, V defaultValue)	Returns the value to which the specified key is mapped, or defaultValue if this map contains no mapping for the key.
int	hashCode()	Returns the hash code value for this map.
boolean	isEmpty()	Returns true if this map contains no key-value mappings.
Set<K>	keySet()	Returns a Set view of the keys contained in this map.

Bag - Primera parte

b. Para practicar los mensajes investigados anteriormente, escriba un test de unidad que contenga lo siguiente:
    a. Cree un map un `Map<String, Integer>`, y agregue las tuplas (“Lionel Messi”, 111), (“Gabriel Batistuta”, 56), (“Kun Agüero”, 42)
    b. Elimine la entrada con clave “Kun Agüero”
    c. Messi hizo 112 goles a día de la fecha; actualice la cantidad de goles
    d. Intente repetir la clave “Gabriel Batistuta” y verifique que no es posible.
    e. Obtenga la cantidad total de goles
c. Como se mencionó, cualquier objeto puede actuar como clave. Es decir, pueden ser instancias de clases definidas por el programador. Modele e implemente la clase Jugador con apellido y nombre. Escriba otro test de unidad similar al de la tarea 2, pero utilizando `Map<Jugador, Integer>`

Bag - Primera parte

b. Para practicar los mensajes investigados anteriormente, escriba un test de unidad que contenga lo siguiente:
   a. Cree un map un `Map<String, Integer>`, y agregue las tuplas ("Lionel Messi", 111), ("Gabriel Batistuta", 56), ("Kun Agüero", 4
   b. Elimine la entrada con clave "Kun Agüero"
   c. Messi hizo 112 goles a día de la fecha; actualice la ca
   d. Intente repetir la clave "Gabriel Batistuta" y verifique q
   e. Obtenga la cantidad total de goles
c. Como se mencionó, cualquier objeto puede actuar como clav
   instancias de clases definidas por el programador. Modele e i
   Jugador con apellido y nombre. Escriba otro test de unidad similar al de la tarea 2, pero utilizando `Map<Jugador, Integer>`



FACULTAD DE INFORMATICA | UNIVERSIDAD NACIONAL DE LA PLATA

Bag - Primera parte

- Lo anterior no es un diseño OO adecuado. Es decir, no sería correcto en términos OO tener un “string” (el nombre del jugador) asociado a una cantidad de goles en una estructura de datos “auxiliar”
- Tampoco sería correcto, desde el punto de vista OO, teniendo una clase jugador, que exista una estructura de datos “auxiliar” que nos diga “cuantos goles hizo un jugador”
- **Lo hacemos únicamente porque es un ejercicio para aprender y practicar usar Map, no estamos diseñando una solución OO. El foco es otro**
- **El diseño OO adecuado sería que exista la clase Jugador y que sepa responder cuántos goles hizo**
  - Podría tener una variable de instancia o una lista de goles y contarlos

Bag - Segunda parte

Un **Bag** (bolsa) es una colección que permite almacenar elementos sin ningún orden específico y admite elementos repetidos.

Este objeto requiere un buen tiempo de respuesta para conocer la **cardinalidad** de sus elementos, y por esa razón almacena la cardinalidad de cada elemento (cantidad de veces que fue agregado en la bolsa). Por ejemplo, si agregamos 3 veces un objeto en la bolsa, y luego eliminamos 1 referencia, la cardinalidad de ese objeto en la bolsa es 2.

Pan	1
Leche	1

Al agregarlo en el bag, suma 1

FACULTAD DE INFORMATICA | UNIVERSIDAD NACIONAL DE LA PLATA

```java
public interface Bag<T> extends Collection<T> {
    /**
     * Agrega un elemento al Bag, incrementando en 1 su cardinalidad.
     */
    @Override
    boolean add(T element);

    /**
     * Devuelve la cardinalidad del elemento. Sí el elemento no está en el Bag,
     * devuelve 0.
     */
    int occurrencesOf(T element);

    /**
     * Elimina una referencia del elemento del Bag. Sí el elemento no está en el
     * Bag, no hace nada.
     */
    void removeOccurrence(T element);

    /**
     * Elimina el elemento del Bag. Sí el elemento no está en el Bag, no hace
     * nada
     */
    void removeAll(T element);

    /**
     * Devuelve el número total de elementos en el Bag, es decir, la suma de
     * todas las cardinalidades de todos sus elementos.
     */
    @Override
    int size();
}
```

Observe que la interfaz Bag<T> extiende Collection<T>.

Bag - Segunda parte

Tareas:

1. Liste los métodos que debe contener una clase que implementa la interface Bag<T>.
2. Explique cómo implementaría un **Bag<T>** usando composición con un **Map<K, V>**. ¿De qué tipo tendrían que ser las claves y valores del Map?
3. Implemente la interfaz Bag<T>, utilizando **AbstractCollection<T>** como superclase, y componga con un **Map<T, V>**. Para simplificar la implementación utilice la clase BagImpl que se encuentra en este link.

4. Discuta con un ayudante:
    a. ¿Cuáles son los beneficios de utilizar **AbstractCollection** como superclase para implementar el Bag?
    b. ¿Qué ventajas tiene componer con un Map para implementar el Bag?
    c. En lugar de componer con un Map, ¿es posible extenderlo para poder implementar el Bag? ¿Qué diferencias tendría esa solución con respecto a la planteada en este ejercicio?
    d. ¿Para qué cree que podría ser útil un objeto Bag?

Ejercicio 22. Estadísticas del Cliente de Correo.

Extienda el Ejercicio 13: Cliente de Correo:

Nos piden implementar la siguiente funcionalidad:

- cantidad de emails que tiene una carpeta
- cantidad total de emails en el cliente de correo: considerando todas las carpetas existentes.
- cantidad de mails por categoría: para cada carpeta se debe calcular y retornar en un solo objeto, la cantidad de emails categorizados por tamaño siguiendo el siguiente criterio
    - Pequeño: el email tiene un tamaño entre 0 y 300
    - Mediano: el email tiene un tamaño entre 301 y 500
    - Grande: el email tiene un tamaño mayor a 501

Ejercicio 23: Mercado de objetos

Queremos programar en objetos una versión simplificada de un mercado on-line similar a e-Bay o MercadoLibre. En esta plataforma, los vendedores registrados pueden publicar productos disponibles a la venta, y los clientes pueden realizar pedidos para comprar esos productos.

Nos piden implementar la siguiente funcionalidad:

Crear un pedido para un cliente: dado un cliente, una forma de pago, una forma de envío, un producto y la cantidad solicitada, se verifica sí hay unidades disponibles del producto: sí es así, se crea el pedido y se descuentan las unidades indicadas del producto. No hace nada si no hay suficientes unidades disponibles.

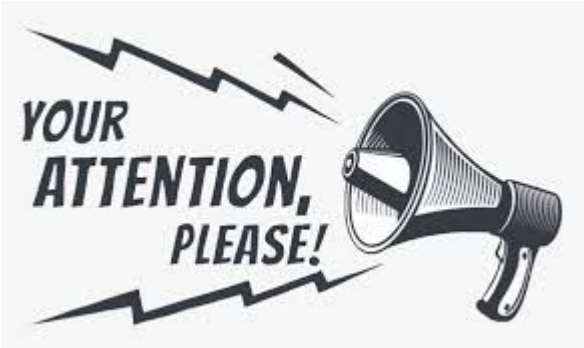
Conocer la cantidad de productos pedidos por categoría: dado un cliente, se desea conocer cuántos productos por categoría ha pedido en la plataforma. Ejemplos de categorías son: “Electrodomésticos”, “Computadores”, “Hogar”, etc.

Ejercicio 23: Mercado de objetos

Calcular el costo total de un pedido. Dado un pedido, se retorna su costo total que se calcula de la siguiente forma: (precio final en base a la forma de pago seleccionada) + (costo de envío en base a la forma de envío seleccionada).

- Si la forma de pago es "al contado", el precio final es el que se indica en el producto
- Si la forma de pago es "6 cuotas", el precio final se incrementa en un 20%
- Si la forma de envío es "retirar en el comercio" no hay costo adicional de envío.
- Si la forma de envío es "retirar en sucursal del correo" el costo es de $3000.
- Si la forma de envio es "express a domicilio" el costo es $0.5 por Km de distancia entre la dirección del vendedor y del cliente. Asuma que existe una clase CalculadoraDeDistancia, cuyas instancias entienden el mensaje #distanciaEntre que recibe dos direcciones y retorna la distancia en Km entre ellas. Por ahora trabaje con una implementación propia para pruebas que siempre retorna 100 (sin importar las direcciones recibidas).

Ejercicio 23: Mercado de objetos



- **Tests de unidad**

  Tareas:

  a) **Modele e implemente**
     i) Diagrama de clases UML.
     ii) Implemente en Java la funcionalidad requerida.
  b) **Pruebas automatizadas**
     i) Diseñe los casos de prueba teniendo en cuenta los conceptos de valores de borde y particiones equivalentes vistos en la teoría.
     ii) Implemente utilizando JUnit los tests automatizados diseñados en el punto anterior.

Ejercicio 23: Mercado de objetos

- **¿Qué varía al calcular la forma de pago del costo de un pedido?**

  Contado —> Es el costo del pedido

  6 cuotas —> Se incrementa un 20%

Ejercicio 23: Mercado de objetos

- **¿Qué varía al calcular el mecanismo de envío del costo de un pedido?**

  Retiro en comercio —> Es el costo del pedido

  Retiro en correo —> $3000

  Express a domicilio —> $0.5 por Km de distancia

Ejercicio 23: Mercado de objetos

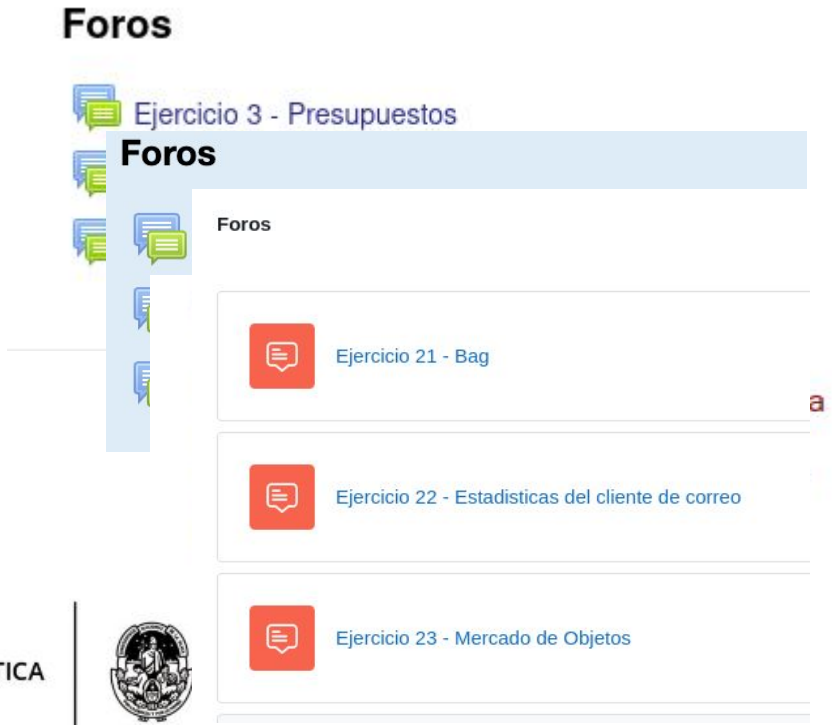
- **¿Cuáles particiones equivalentes podemos identificar?**
  - Pago contado
    - Retiro en comercio
    - Retiro en correo
    - Express a Domicilio
    -
  - 6 Cuotas
    - Retiro en comercio
    - Retiro en correo
    - Express a Domicilio

Foros de consulta

Cómo preguntar en el foro

Breve guía para poder sacar el mejor provecho al foro y a la convivencia a través de las preguntas y respuestas.

- Cómo preguntar en el foro
  - Antes de Preguntar: Busca una respuesta por tus propios medios
  - Elegí el foro específico
  - Elegí un título apropiado para la pregunta
  - No envíes una solución para que la corrijan
  - Describir qué estás intentando hacer
  - Describir el problema y lo que has intentado para resolverlo
  - Escribir claro
  - No solicites respuestas a tu correo
  - Si no entendés la respuesta
  - Terminá con una breve nota de conclusión.
  - Evitá el “Me sumo al pedido”



FACULTAD DE INFORMATICA