ADSOF: Convocatoria Ordinaria – Evaluación Contínua

Ejercicio 3 v1 (2.5 p)

Solución:

Huecos: (1): nada (2): try (3): catch(ItemNotFoundException exc) ➔**0,25p**

Total:0,5p

```
interface IValuable<T extends Comparable<T>> extends Comparable<IValuable<T>>{    ➔ (requisitos sobre T e IValuable) 0,25p
        T getValue();
        default String getDesc() { return toString(); }
        @Override default int compareTo(IValuable<T> o) { return getValue().compareTo(o.getValue()); } ➔ métodos default 0,25p
}
```

```
class Item implements IValuable<Long>{
        private String name;
        private Long   initPrice;
        public Item (String i, Long price) {
                this.name = i;
                this.initPrice = price;
        }
        @Override public String toString() { return this.name; }
        @Override public Long getValue() { return initPrice; }
}
```

Total:0,25p

(Errores comunes:
-- Item genérica cuando no lo es
-- Item no genérica, pero usa un parámetro generico
-- Añadir a Item las pujas o las condiciones (es mejor añadir esto en Auction, ya que de otro modo habría que reimplementarlo en cada clase que implemente IValuable)

```
class ItemNotFoundException extends Exception {
        public ItemNotFoundException(String item) {
                super("Item "+item+" not found");
        }
}
```

Total:0,25p

(Errores comunes:
-- Almacenar aquí un Item, que haría esta excepción dependiente de Item. Esto es un error, ya que Auction es independiente de Item, que sólo es un ejemplo de clase que implementa IValuable.)

Total: 1,25p

```
public class Auction<T extends Comparable<T>, I extends IValuable<T>> {   ➔ Requisitos sobre T e I: 0,2
        private Map<I, SortedSet<T>> bids = new TreeMap<>();               ➔ 0,2
        private Map<I, Predicate<T>> sellConditions = new HashMap<>();     ➔ 0,2

        public void addItem(I item, Predicate<T> pred) {                   ➔ 0,2
                this.bids.put(item, new TreeSet<>() );
                this.sellConditions.put(item, pred);
        }

        public boolean bid(String item, T p) throws ItemNotFoundException{        ➔ 0,45 (y métodos auziliares)
                I bidItem = this.itemDesc(item);
                SortedSet<T> itemBids = this.bidsFor(item);
                if (itemBids==null) throw new ItemNotFoundException(item);
                if (!highestBid(itemBids, p, bidItem)) return false;
                itemBids.add(p);
                if (this.sellConditions.get(bidItem).test(p))
                        return true;
                return false;
        }

        private boolean highestBid(SortedSet<T> itemBids, T bid, I it) {
                if (itemBids.size()==0) return it.getValue().compareTo(bid)<0;
                if (itemBids.tailSet(bid).size()>0) return false;//no the highest bid so far
                return true;
        }

        @Override public String toString() { return this.bids.toString();         }       ➔-0,1 si no está

        private I itemDesc(String desc) {
                for (I itm : this.bids.keySet())
                        if (itm.getDesc().equals(desc)) return itm;
                return null;
        }

        private SortedSet<T> bidsFor(String itemDesc) {
                I item = this.itemDesc(itemDesc);
                if (item == null) return null;
                return this.bids.get(item);
        }
}
```

(continúa detrás)

Algunos comentarios sobre la solución:

- **Huecos:**
  - En (1) no hay que poner nada, ya que la excepción se captura
- **Interfaz IValuable:**
  - T e IValuable deben ser comparables
  - Añadimos métods default (compareTo, getDesc) para facilitar su implementación
- **Clase Item:**
  - Item no es genérica, como muestra el main
  - Item debe implementar IValuable<Long>
  - Si IValuable no fuese comparable, Item debería implementar Comparable e implementar los métodos. Por tanto, es mejor que IValuable nos proporcione ya ese soporte.
  - Declarar aquí atributos y métodos para las pujas o los predicados no es óptimo, ya que habría que reimplementarlo en clases que implementen IValuable. Auction no tiene dependencias con Item, sino con IValuable.
- **Clase excepción ItemNotFoundException**:
  - Debe extender de Exception
  - No puede extender de RuntimeException, porque no habría manera de rellenar razonablemente los huecos del main
  - Tiene que tener ese nombre por lo que se imprime en consola.
  - auctionsSimple es el nombre del paquete, no es el nombre de la clase
- **Clase Auction:**
  - Tiene dos parámetros genéricos: uno debe ser comparable y otro extender de IValuable.
  - Es un error muy grave de concepto poner en los parámetros genéricos cosas como Item, Double o similar, ya que son nombres de tipos concretos
  - Declarar que una clase lanza una excepción es un error grave de concepto. Las excepciones las lanzan los métodos.
  - Las pujas han de asociarse a los elementos, y esto deben ordenarse en orden de salida. Por esto, lo óptimo es usar un mapa, donde el orden natural sea el precio de salida. Las pujas debe ordenarse por orden ascendente, por que la estructura más apropiada sería un conjunto ordenado. Utilizar como clave del mapa un String no es válido, ya que el criterio de orden no se satisface.
  - Han de asociarse predicados a los elementos, para ello hace falta un mapa (aunque TreeSet es innecesario).
  - Errores comunes en método bid: no declarar que lanza la excepción, no lanzarla, no comprobar que es la puja más alta, no guarder la puja
- **Otros errores comunes**
  - Usar clases o interfaces genéricas sin parámetros
  - No poner control de acceso private a los atributos (error básico de concepto)
  - Poner un constructor con 0 parámetros sin código (por ejemplo en Auction) es innecesario.
  - Tratar de copiar de tus compañeros es siempre un error