3. Resumen Laboratorio Excepciones

Programación Modular y Orientación a Objetos

Dpto. de Lenguajes y Sistemas Informáticos
UPV/EHU

Clase Product: método setAmount

```
public void setAmount(int amount) throws NegativeAmountException{
   if(amount<0)
     throw new NegativeAmountException(“Wrong value");
   else
     this.amount = amount;
}
```

Se lanza una excepción si el valor es negativo

```
throw new NegativeAmountException(“Wrong value");
```

Como no se trata, se indica que el método puede elevarla

```
throws NegativeAmountException
```

Clase Stock: método updateAmount

```
public void updateAmount(int code, int amount)throws
NegativeAmountException,UnknownCodeException {
        Product p = new Product(code);
        int pos = list.indexOf(p);
        if (pos != -1) {
                list.get(pos).setAmount(amount);
        }else {
                throw new UnknownCodeException("Not valid code");
        }
}
```

El método setAmount puede elevar la excepción **NegativeAmountException,** como no se trata en el método, se debe indicar que el método puede elevarla

```
throws NegativeAmountException
```

Se puede lanzar la excepción **UnknownCodeException**

```
throw new UnknownCodeException("Not valid code");
```

Como no se trata, se indica que el método puede elevarla

```
throws UnknownCodeException
```

Clase UnknownCodeException

- Las excepciones se pueden definir como una clase externa o una clase interna a otra

```
public class Stock{
//Atributos, constructoras y métodos
...
    public class UnkownCodeException extends Exception{
                public UnkownCodeException(){super();}
                public UnkownCodeException(String s){super(s);}
     }
}
```

Clase SuperOnline

Opción 1

```
try {
        ins.updateAmount(1,10);
} catch (Stock.unknownCodeException | NegativeAmountException e) {
        System.out.println("Not valid instruction");
}
```

Opción 2

```
try {
        ins.updateAmount(1,10);
} catch (Stock.unknownCodeException e) {
        System.out.println("Unknown code");
} catch (NegativeAmountException e) {
        System.out.println("Negative amount");
}
```

Clase SuperOnline

- Si queremos que se ejecuten todas las apariciones de updateAmount, se pone cada llamada en un bloque try-catch.
- Si queremos que cuando fracase una llamada el resto no se ejecuten, pondremos todas las llamadas en un únio bloque try-cathc

Escritura en ficheros

```
public void storeStockInFile() {
  FileWriter fs;
  try {
    fs = new FileWriter("salida.txt");
    fs.write(toString());
    fs.close();
  } catch (IOException e) {
    e.getMessage();
  }
}
```