UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escola Tècnica Superior d'Enginyeria Informàtica etsinf

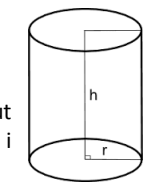
Exercicis Tema 3

Programació (PRG)
Curs 2015/16

Departament de Sistemes Informàtics i Computació

- **Descarrega** (del Tema 3 de PoliformaT) el fitxer ***exercicisT3.zip*** i extrau el seu contingut (carpeta ***exercicisT3***) en la carpeta ***prg*** dins del teu ***disc W***
- Cadascun dels exercicis que se't proposen a continuació es correspon amb un dels fitxers ***.jar*** de la carpeta ***exercicisT3***. Des de l'opció **Projecte** de ***BlueJ***, usa l'opció **Obre No BlueJ** per tal d'obrir-los com projectes ***BlueJ*** i prepara't per usar-los.

Exercici: classe Cilindre



1. A la classe `Cercle`, quin modificador de visibilitat hauria de tenir l'atribut `radi` perquè siga directament accessible des de la classe `Cilindre` i afavorisca el principi d'ocultació d'informació?

2. De les següents implementacions del constructor de `Cilindre`, una d'elles és incorrecta, quina? Per què?

   a)
   ```
   super(radiBase, color);
   super.tipus = "Cilindre";
   altura = alt;
   ```

   b)
   ```
   super.color = color;
   super.radi = radiBase;
   super.tipus = "Cilindre";
   altura = alt;
   ```

3. A la classe `Cilindre`, modifica les implementacions dels mètodes `getArea()`, `getVolum()` i `toString()` per afavorir el principi de reutilització de software.

BlueJ:actorPelicules Exercici: Actors i Pel.lícules - Errors

- Corregeix el codi de les classes `Persona`, `Actor` i `Pelicules` per tal que el mètode `mostrarRepartiment` siga correcte sintàctica i semànticament:

```
public class Persona {
    private String nom;
    public Persona(String n) {
        this.nom = n;
    }
}
```

```
public class Actor extends Persona {
    private String pelicula;
    public Actor(String n, String p) {
        this.nom = n;
        this.pelicula = p;
    }
}
```

```
public class Pelicules {
  public static void mostrarRepartiment(Actor[] llista, String pelicula){
      for (int i = 0; i <= llista.size; i++) {
          if (llista[i].pelicula == pelicula) {
              System.out.println(llista[i].toString());
          }
      }
  }
}
```

Exercici cortesia d'Oscar Sapena

BlueJ:bonoMetro Exercici: BonoMetro amb Excepcions

- La classe `BonoMetro` permet representar el títol de transport amb el que se pot viatjar al metro.
  - Cada bonometro té un nombre de tiquets disponibles i tots els bonometros comparteixen una quantitat per defecte de recàrrega que és 10.
  - Un bonometro es pot carregar amb un nombre donat de viatges o per defecte; es pot consultar el nombre de viatges disponible i es pot recarregar.
  - El mètode **`picar`** comprova si queden viatges, en aquest cas actualitza el nombre de tiquets i torna el missatge `"Bono amb xxx tiquets"`. Si no queden viatges torna el missatge `"Bono esgotat. Recàrrega'l JA!"`

1. Modifica el mètode `picar` per tal que quan no queden viatges llance l'excepció `SaldoEsgotatException`.
2. Revisa la implementació de l'excepció d'usuari `SaldoEsgotatException`. Fixa't que és una subclasse d'`Exception` i, per tant, ha de tractar-se com una excepció *checked*.
3. Completa la classe `TestBonoMetro` per tal que verifique la funcionalitat prèviament implementada.

BlueJ:carnetPerPunts

Exercici: Carnet per Punts

- La classe `CarnetDeConduir` permet representar un carnet de conduir amb un crèdit inicial de 12 punts que es va perdent a mesura que es cometeixen infraccions. Un saldo de zero punts o negatiu implica una retirada immediata del carnet de conduir.
- La classe `DGT` permet aplicar la penalització.

1. Revisa l'excepció *comprovada* d'usuari `RetiradaImmediataCarnet`.
2. Modifica el disseny actual del mètode `llevarPunts` de la classe `CarnetDeConduir` per tal que, quan el saldo de punts d'un carnet de conduir siga negatiu o zero després de la penalització, llance l'excepció `RetiradaImmediataCarnet`.
3. Modifica el mètode `multar` de la classe `DGT` per a que mostre un missatge d'error per pantalla si la penalització de punts comporta la retirada immediata del carnet.

BlueJ:transferirFitxers

Exercici: Transferència de Fitxers

- La classe `CopyViaFTP` permet realitzar la transferència d'un arxiu a un altra màquina mitjançant FTP. Quan, pel motiu que siga, aquesta transferència no es pot fer, llança l'excepció comprovada d'usuari `UnableToTransferException`.

```
public class CopyViaFTP {
    public static void copyTo(String hostName, String localFilePath)
        throws UnableToTransferException {
        …
    }
}
```

1. Completa el main de la classe `TestCopyViaFTP` per tal que faça la transferència del fitxer `/tmp/data` a la màquina `fileserver.upv.es`. En cas d'error, l'operació s'haurà de reintentar un màxim de 3 vegades i indicar a l'usuari el número d'intent.

BlueJ:modulAutoritzacio

Exercici: Mòdul d'Autorització

- La classe `AuthModule` implementa un mòdul d'autorització basat en usuaris i contrassenyes registrats:

```
public static void check(String username, String password)
    throws InvalidUserException, InvalidPasswordException,
    ExpirationDeadlineException
```

- Excepcions llançades:
  - `InvalidUserException`: Si el nom d'usuari no existeix.
  - `InvalidPasswordException`: El nom d'usuari existeix, però la contrassenya no coincideix amb la registrada en el sistema.
  - `ExpirationDeadlineException` (`extends RuntimeException`): La contrassenya caducarà en breu.

1. A la classe `GrantAccess`, utilitzant la classe `AuthModule`, completa el mètode `grantAccess` que mostra per l'eixida estàndard un missatge indicant l'autorització per accedir al sistema a un usuari, de nom i contrassenya donats, o el motiu pel qual no s'ha concedit l'acces.
   - Ha de llançar l'excepció `AccessDeniedException` en cas que el nom d'usuari no existisca o la contrassenya no coincidisca amb la registrada.