Grau d'Enginyeria Informàtica. Facultat de Matemàtiques i Informàtica. UB
Curs 2024-2025

Pràctica 3: Part 1

Daniel Ortiz

1 Introducció

En aquesta pràctica aplicarem el que s'ha après a l'assignatura per implementar un sistema de simulació d'un reactor d'aigua pressuritzada. A diferència de les pràctiques anteriors, la Pràctica 3 es desenvoluparà en dues parts. A la part 1 de la pràctica, s'implementarà una aplicació seguint el paradigma de programació orientada a objectes que modeli el funcionament d'un reactor d'aigua pressuritzada. En aquesta aplicació, la vista serà implementada com una interfície de text. A la part 2 de la pràctica, es proposa reemplaçar la vista implementada a la part 1 per una interfície gràfica, aplicant conceptes de programació orientada a esdeveniments. Aquest document descriu la part 1 del treball a realitzar.

2 Descripció del Problema

Se'ns ha demanat desenvolupar un sistema de simulació d'un reactor d'aigua pressuritzada. Un reactor d'aigua pressuritzada és un tipus de reactor nuclear que fa servir aigua a alta pressió per a produir vapor que després es transforma en energia elèctrica. Com a curiositat, aquesta tecnologia es fa servir a les centrals nuclears Ascó I i II i Vandellòs II a la província de Tarragona.

L'objectiu de la pràctica és continuar treballant amb conceptes de programació orientada a objectes al mateix temps que s'implementa un simulador d'una central nuclear extremadament simplificat. Aquest simulador també es pot veure com a un senzill joc on les decisions que es prenen tenen una recompensa o penalització en forma de puntuació que es va acumulant al llarg d'una partida.

2.1 Funcionament i Components de la Central

Una central nuclear d'aigua pressuritzada fa servir un reactor de fissió nuclear per calentar aigua que recorre un circuit denominat circuit primari. Aquest circuit està pressuritzat, permetent que l'aigua arribi a temperatures molt altes en estat líquid. L'aigua calenta del reactor circula pel circuit primari gràcies a unes bombes refrigerants que constitueixen el sistema de refrigeració de la central. L'aigua calenta passa per un component de la central denominat generador de vapor, que transmet el calor de l'aigua del circuit primari, a un altre circuit denominat circuit d'aigua secundari. El circuit secundari està totalment separat del primari i l'aigua que conté no està pressuritzada. Al calentar-se, l'aigua del circuit secundari es converteix en vapor que mou les aspes d'una turbina, generant energia elèctrica. Després d'això, el vapor es torna a convertir en aigua líquida mitjançant un condensador.

Pots consultar la Figura 1 per tenir una visió general del procés[1]. A continuació teniu un resum dels conceptes il·lustrats a la figura:

- **Barres de control**: es fan servir per controlar la intensitat de la reacció nuclear. Per a això, es poden inserir dins del combustible. Per exemple,

[1] Una representació una mica més detallada i amb una animació es pot trobar en aquest enllaç.

si s'insereixen al 100%, la reacció nuclear s'atura. En canvi, si s'extreuen completament (es a dir, el grau d'inserció és del 0%), la reacció es produeix amb la màxima intensitat.

- **Reactor**: fa servir combustible radioactiu per generar energia en forma de calor que es transmet a l'aigua del circuit primari.
- **Sistema de refrigeració**: el sistema de refrigeració es composa d'una serie de bombes refrigerants que mouen l'aigua calenta generada pel reactor fins al generador de vapor. D'aquest manera, el sistema de refrigeració extreu el calor generat pel reactor, i el transporta fins al generador de vapor.
- **Generador de vapor**: el generador de vapor transmet calor del circuit d'aigua primari al circuit d'aigua secundari. Això permet que l'aigua en estat líquid del circuit secundari es converteixi en vapor. A més, quan es transmet calor del circuit primari al secondari, la temperatura de l'aigua disminueix.
- **Turbina**: el vapor generat pel generador es fa servir per moure la turbina, generant energia elèctrica.
- **Condensador**: el condensador es fa servir per convertir el vapor del circuit secundari de nou en aigua líquida, completant el cicle.

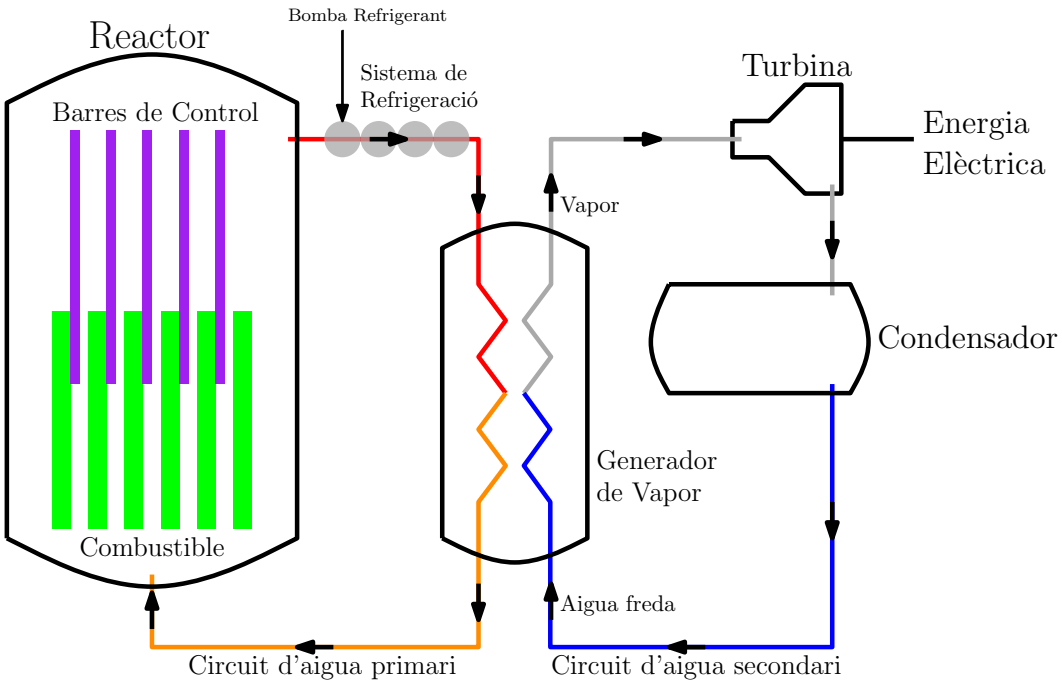

Figura 1: Esquema de funcionament d'un reactor d'aigua pressuritzada.

2.2 Generació d'Energia Elèctrica

La generació d'energia elèctrica dins de la central nuclear es pot veure com el resultat de la concatenació del components de la central, de manera que cada component rep l'input del component anterior i genera un output que es connecta amb el component següent de la manera que es detalla a continuació:

1. **El reactor** rep com a input el grau d'inserció de les barres de control en percentatge i genera com a output una determinada quantitat de graus Celsius que es transmet a l'aigua del circuit primari.

2. **El sistema de refrigeració** té com a input el calor generat pel reactor i genera com a output la quantitat de calor que és capaç de transportar des del reactor fins al generador de vapor. Aquesta quantitat depèn de la quantitat de bombes refrigerants que estiguin activades. **A la nostra central hi haurà un total de 4 bombes refrigerants**.

3. **El generador de vapor** transmet el calor transportat pel sistema de refrigeració (input) fins al circuit secundari (output) amb una determinada eficiència, fixada en 0.9 al nostre cas. Es considera 25° la temperatura ambient.

4. **La turbina** converteix els graus del generador de vapor en unitats de potència. Això només es podrà fer si la temperatura generada com a output pel generador de vapor es superior a 100 graus (la temperatura de vaporització de l'aigua).

Cadascun dels components de la central pot estar activat o no.

La Figura 2 mostra un resum del procés de generació d'electricitat per part dels components de la central d'aigua pressuritzada. A més la figura inclou com es calcula l'output de cada component a partir de l'input. Aquest càlcul ha de tenir en compte si els components estan activats o no.

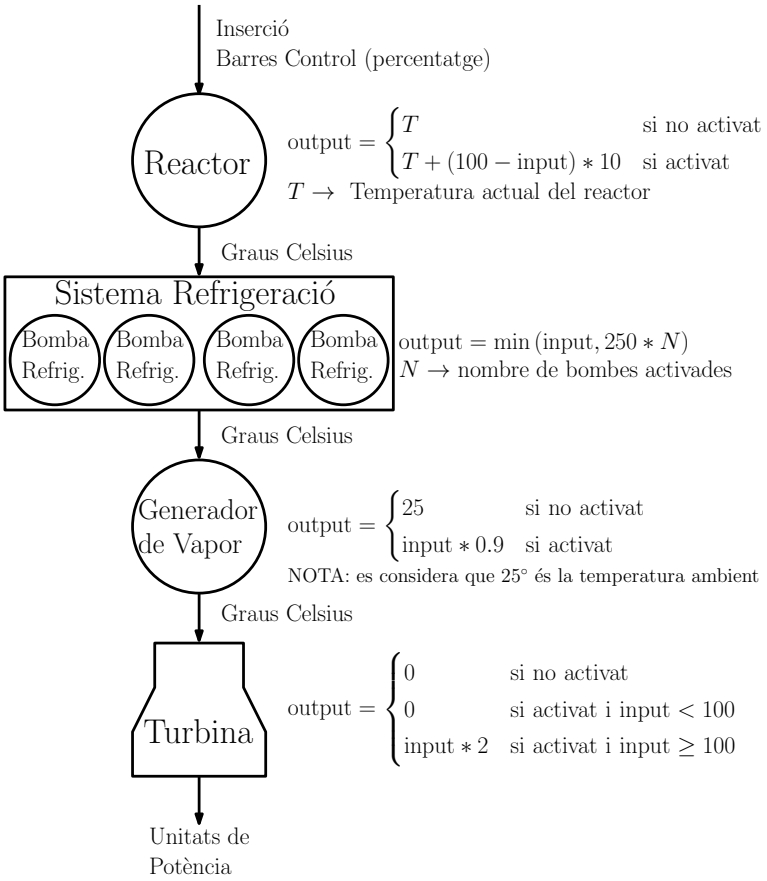

Figura 2: Inputs i outputs dels components de la central.

2.3 Actualització al Finalitzar un Dia

El simulador que s'ha d'implementar funciona per torns que engloben dies complets. A l'inici de cada dia es dona una demanda de potència que s'ha de satisfer i l'operador de la central ha d'ajustar el funcionament dels components per generar aquesta potència. **Quan l'operador està satisfet amb la configuració del components, pot donar el dia per finalitzat**. Això es farà mitjançant una opció específica dins de la interfície del simulador. **Quan el dia acaba, s'ha d'actualitzar la informació econòmica amb la configuració de la central que s'ha establert. Després d'això, s'actualitzarà l'estat de la central. Aquest actualització inclou la refrigeració del reactor i la revisió dels components de la central, determinant el seu estat per al següent dia. A més, la informació derivada de les decisions de l'operador de la central es registrarà dins d'un document específic (bitàcola)**.

2.3.1 Actualització de l'Economia de la Central

Al finalitzar un dia, s'ha d'actualitzar la informació econòmica de la central. Per això, es tindrà en compte la seva configuració actual. En concret, s'ha d'obtenir la següent informació:

1. **Beneficis per potència generada**: cada unitat de potència satisfeta proporcionarà un benefici d'una unitat monetària. Per exemple, si es demanaven 100 unitats de potència i es van generar 80, els beneficis seran de 80 unitats econòmiques. En canvi, si es van generar 120, els beneficis seran de 100 unitats.

2. **Penalització per excés de potència**: si s'excedeix la demanda de potència per al dia en curs, s'incurrirà en una penalització de 250 unitats econòmiques.

3. **Costos operatius**: per cada component actiu, s'haurà de sumar el seu cost operatiu en unitats monetàries segons la Taula 1.

4. **Guanys acumulats**: s'obtenen restant la penalització i els costos operatius als beneficis, i després sumant la quantitat resultant als guanys anteriors.

2.3.2 Actualització de l'Estat de la Central

L'actualització de l'estat de la central al finalitzar el dia inclou la refrigeració del reactor i la revisió de l'estat dels components.

Refrigeració del Reactor La nova temperatura s'obté calculant l'output del reactor i restant la temperatura extreta pel sistema de refrigeració. Per exemple, si no hi ha cap bomba refrigerant encesa, el reactor no es refrigera i al següent dia tindrà com a temperatura la temperatura produïda com a output. **La temperatura del reactor mai podrà ser inferior a 25 graus, que es considera el valor de la temperatura ambient**.

Revisió de l'Estat del Components Al finalitzar el dia, s'ha de revisar l'estat dels components de la central. Al revisar un component determinat, poden ocórrer incidències que facin que quedi inoperatiu. A conseqüència d'això, el component

es desactivarà. A més, un component afectat per una incidència no permetrà que s'executin determinades accions. La Taula 1 mostra els detalls.

	Cost Operatiu	**Incidències**	**Accions no Permeses**
Barres de Control	5	-	No es permet fixar un grau d'inserció fora de l'interval 0-100
Reactor	35	Si la temperatura supera 1.000 graus, s'ha de desactivar automàticament el reactor.	No es pot activar el reactor mentre es superi la temperatura màxima de 1.000 graus.
Sistema de Refrigeració	130xBomba	Al finalitzar un dia hi ha una probabilitat del 25% que cada bomba refrigerant es quedi fora de servei per al dia següent. Una bomba fora de servei s'ha de desactivar automàticament.	No es pot activar una bomba refrigerant si està fora de servei.
Generador de Vapor	25	-	-
Turbina	20	-	-

Taula 1: Informació addicional sobre els components de la central. **El cost operatiu de qualsevol component no activat es zero**.

2.3.3 Registre d'Informació de la Central

La informació relacionada amb les actualitzacions anteriors s'haurà de registrar en un quadern de bitàcola que s'explica a l'apartat següent.

2.4 Quadern de Bitàcola

En una central nuclear és necessari portar un registre molt estricte del seu funcionament. Amb aquest propòsit, s'ha de mantenir un quadern de bitàcola, que es composa d'una sèrie de pàgines que poden ser de tres tipus (d'estat, econòmica i d'incidències). Totes les paginés tenen en comú que incorporen el dia a què es refereixen. **Al finalitzar cada dia, es generarà una pàgina econòmica, una pàgina d'estat i una pàgina d'incidències**.

2.4.1 Pàgina Econòmica

Les pàgines econòmiques tenen el següent contingut:

- Nombre de dia al que es refereix la pàgina.
- Demanda de potència elèctrica.
- Potència generada.
- Percentatge de la demanda de potència satisfeta.
- Beneficis, penalització per excés de potència, costos operatius i guanys acumulats segons s'han descrit a l'Apartat 2.3.1.

2.4.2 Pàgina d'Estat

Les pàgines d'estat contenen la següent informació:

- Nombre de dia al que es refereix la pàgina.
- Grau d'inserció de les barres de control.
- Output dels components de la central (reactor, sistema de refrigeració, generador de vapor i turbina).

2.4.3 Pàgina d'Incidències

Per últim, les pàgines d'incidències contenen el següent:

- Nombre de dia al que es refereix la pàgina.
- Una llista de cadenes de caràcters, cadascuna descrivint possibles incidències. Dins del mateix dia hi pot haver ninguna, una o més d'una incidència.

3 Objectiu i Funcionalitats

Es pretén crear un sistema d'informació que simuli un reactor d'aigua pressuritzada en què es recullin les especificacions que s'han detallat en la secció anterior. Amb aquest propòsit, igual que en anteriors pràctiques, seguirem un paradigma de programació orientada a objectes.

L'aplicació a desenvolupar ha d'oferir les següents funcionalitats:

1. **Gestió Barres de Control**: Dona pas a un submenú que permet obtenir o establir la inserció de les barres, que ha de ser un nombre real entre 0 i 100.
    1. **Obtenir Inserció Barres**: Mostra per pantalla la inserció de les barres.
    2. **Establir Inserció Barres**: Sol·licita a l'usuari el grau d'inserció de les barres.
    3. **Sortir**: Torna al menú principal.
2. **Gestió Reactor**: Mostra un sub-menú per gestionar el reactor.
    1. **Activar Reactor**: Permet activar el reactor.
    2. **Desactivar Reactor**: Permet desactivar el reactor.
    3. **Mostrar Estat**: Mostra si el reactor està activat i la seva temperatura.
    4. **Sortir**: Torna al menú principal.
3. **Gestió Sistema Refrigeració**: Mostra un sub-menú amb opcions per controlar el sistema de refrigeració.
    1. **Activar Totes les Bombes**: Activa totes les bombes refrigerants.
    2. **Desactivar Totes les Bombes**: Desactiva totes les bombes refrigerants.
    3. **Activar Bomba**: Donat el seu identificador numèric (entre 0 i 3), permet activar una bomba refrigerant.

   4. **Desactivar Bomba**: Donat l'identificador numèric d'una bomba refrigerant, permet desactivar-la.
   5. **Mostrar Estat**: mostra l'estat actual de totes les bombes del sistema de refrigeració.
   6. **Sortir**: Torna al menú principal.

4. **Mostrar Estat Central**: Mostra la pàgina de bitàcola d'estat corresponent al dia actual (veure Apartat 2.4.2). **Aquesta informació és provisional i només es farà efectiva al finalitzar el dia**.
5. **Mostrar Bitàcola**: Mostra tot el contingut de la bitàcola fins al dia actual, incloent las pàgines d'estat, econòmiques i d'incidències.
6. **Mostrar Incidències**: Mostra totes les pàgines d'incidències de la bitàcola fins al dia actual.
7. **Obtenir Demanda Satisfeta amb Configuració Actual**: Mostra la demanda de poténcia del dia en curs, la potència generada amb la configuración de la central actual i el percentatge de demanda satisfeta corresponent.
8. **Finalitzar Dia**: Es duen a terme totes les acciones relacionades amb la finalització d'un dia (veure Apartat 2.3).
9. **Guardar Dades**: Guarda les dades de l'aplicació.
10. **Carrega Dades**: Carrega les dades de l'aplicació.
11. **Sortir**: Surt de l'aplicació.

4 Material per a la Pràctica

Per a aquesta pràctica es proporcionen les següents classes i interfícies[2]:

- `Menu`: la classe `Menu` proporciona codi per implementar el sistema de menús que ha d'oferir la aplicació.
- `InDades`: interfície que ha d'implementar la classe Dades.
- `Dades`: es proporciona una plantilla per implementar aquesta classe, que contindrà les dades de l'aplicació. En particular, el fitxer defineix algunes constants, el constructor de la classe i el mètode `finalitzaDia`. A més, també es proporcionen tres mètodes de servei que s'han de completar: `refrigeraReactor`, `revisaComponents` i `actualitzaEconomia`.
- `InComponent`: interfície que han d'implementar els components de la central.
- `InBombaRefrigerant`: interfície per a la classe `BombaRefrigerant`.
- `InBitacola`: la classe `Bitacola` ha d'implementar aquesta interfície.

[2]NOTA IMPORTANT: s'ha de tenir en compte que el codi base no es pot modificar.

- `CentralUB`: aquesta classe dona accés a tota la funcionalitat de la central nuclear. Es proporciona només el codi amb algunes constants i la funcionalitat necessària per a finalitzar un dia i obtenir de manera aleatòria la demanda de potència elèctrica del següent.

- `VariableUniforme`: aquesta classe es fa servir per determinar aleatoriament si una bomba refrigerant estarà fora de servei al començar un nou dia. Només es declara un objecte d'aquesta classe dins del constructor de la classe Dades i es proporciona com a paràmetre al constructor de la classe `BombaRefrigerant`. La classe `BombaRefrigerant` utilitza el mètode `seguentValor` per generar un nombre aleatori que determinarà si la bomba es queda fora de servei.

- `VariableNormal`: aquesta classe es fa servir dins de la classe `CentralUB` per generar la demanda de potència elèctrica de cada nou dia. Tot el codi necessari està implementat al codi base i per tant no la farem servir directament.

5 Descripció de la Pràctica

A continuació, es detallen una sèrie de passos per a resoldre la pràctica que s'ha proposat. Es recomana seguir aquests passos.

5.1 Creació del projecte

El primer pas serà crear un projecte IntelliJ, el qual s'ha d'anomenar `Cognom1Nom1Cognom2Nom2_P31`, on 1 i 2 fan referència als dos membres de la parella, tenint en compte les consideracions següents:

- La primera lletra de cada part en majúscula i la resta en minúscula.

- S'han d'evitar els accents i caràcters especials com ñ o ç. Per exemple, una estudiant amb nom Dolça Martínez Castaña, hauria de crear un projecte amb el nom `MartinezCastanaDolca`.

- La classe principal s'ha de denominar `IniciadorCentralUB`, i el paquet per defecte `prog2.vista`. En el nom del paquet s'han d'utilitzar també els mateixos criteris anteriors.

- A IntelliJ s'ha d'indicar que la classe principal sigui `prog2.vista.IniciadorCentralUB`.

5.2 Paquets del Projecte

El projecte que hem de desenvolupar està basat en el patró Model-Vista-Adaptador. Haurem de crear una classe adaptador, de manera que la vista no accedirà directament a les dades del model, sinó que ho farà a través d'aquesta classe. En conseqüència, el projecte estarà format per 3 paquets:

- `prog2.vista`: la vista contindrà totes les classes relacionades amb el maneig de menú d'opcions. Entre elles, destaca la classe `CentralUB`, que implementarà el menú d'opcions ofert per l'aplicació.

- `prog2.adaptador`: el paquet adaptador únicament contindrà la classe `Adaptador`, que funcionarà com a mediador entre la vista i el model. La vista només podrà utilitzar aquesta classe per accedir a la informació del model.
- `prog2.model`: el model contindrà totes les classes que modelen les dades que s'han de gestionar dins de l'aplicació. Dins el model, tindrà especial importància la classe `Dades`, que contindrà totes les dades de l'aplicació i portarà a terme totes les accions que afecten els mateixos.

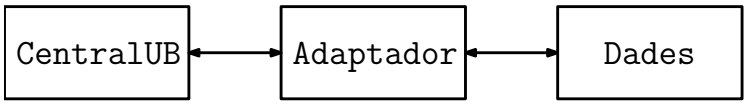

Figura 3: Principals classes del patró Model-Vista-Adaptador i les seves relacions.

A la Figura 3 es mostren les relacions entre les classes principals dels paquets.

5.3 Classes del Projecte

A continuació es descriuen les principals classes a implementar dins del projecte.

Classe `BombaRefrigerant`

La classe `BombaRefrigerant` ha d'implementar la interfície `InBombaRefrigerant`. Cada bomba refrigerant ha de tenir un identificador numèric i dos booleans per indicar si està activada o no i si està fora de servei.

El constructor de la classe ha de rebre l'identificador i un objecte de tipus `VariableUniforme`.

El mètode `revisa` de `BombaRefrigerant` cridarà al mètode `seguentValor` de `VariableUniforme` per a determinar, amb una probabilitat del 25% si la bomba es queda fora de servei durant el dia següent.

Els mètodes `getCapacitat` i `getCostOperatiu` retornaran la capacitat de refrigeració en graus i el cost operatiu, respectivament, de la bomba refrigerant en cas de que estigui activada, o zero en cas contrari.

D'altra banda, la classe també ha de definir el mètode `toString`, que generarà una representació de l'objecte com la següent:

```
Id=0, Activat=false, Fora de servei=false
```

Classes `SistemaRefrigeracio`, `Turbina`, `GeneradorVapor` i `Reactor`

Aquestes classes representen els components de la central i han d'implementar la interfície `InComponent`. El càlcul de l'output de cada component es detalla a la Figura 2. D'altra banda, la Taula 1 incorpora el cost operatiu i les possibles incidències que poden ocórrer per a cada component.

El mètode `revisa` de `InComponent` rep un objecte de tipus `PaginaIncidencies` i s'encarrega de registrar les incidències per a un component. Amb aquest propòsit, es farà servir el mètode `afegeixIncidencia` de la classe `PaginaIncidencies` descrit més avall.

La classe `SistemaRefrigeracio` haurà de contenir un `ArrayList` per emmagatzemar objectes de tipus `BombaRefrigerant` i un mètode `afegirBomba` per afegir-los. El mètode `activa` haurà d'activar totes les bombes (sempre que no estiguin fora de servei), i el mètode `desactiva` s'encarregarà de desactivar-les. El mètode `getActivat` ha de retornar `true` si hi al menys una bomba activada. El mètode `revisa` de `SistemaRefrigeracio` haurà de cridar al mètode del mateix nom de la classe `BombaRefrigerant`. Per a calcular l'output (mètode `calculaOutput`) i el cost operatiu (mètode `getCostOperatiu`), es faran servir els métodes `getCapacitat` i `getCostOperatiu`, respectivament, de la classe `BombaRefrigerant`.

La classe `Reactor` haurà de tenir un atribut de tipus `float` per registrar la temperatura del reactor, així com el seu *getter* i *setter*.

Tots el components hauran de definir un atribut de tipus `boolean` per guardar si estan activats o no.

NOTA: com a ajuda per implementar aquestes classes, es recomana estudiar el constructor de la classe `Dades`, que es proporciona completament implementat al codi base.

Classe `PaginaBitacola`

Aquesta classe implementa la classe base d'altres tres classes usades per representar pàgines de bitàcola (veure Apartat 2.4). La classe `PaginaBitacola` conté un atribut de tipus enter per guardar el nombre de dia al que es refereix la pàgina. Aquest atribut s'ha d'inicialitzar en el constructor.

Classe `PaginaEconomica`

Aquesta classe també hereta de `PaginaBitacola` i ha de tenir el atributs necessaris segons el que es va a explicar a l'Apartat 2.4.1. El atributs s'han d'inicialitzar usant el constructor. A més, s'han de definir *getters* per a cadascú.

La classe `PaginaEconomica` ha de definir un mètode `toString` que generi una representació de la pàgina com a `String`. Un exemple de la cadena a generar seria[3]:

```
# Pàgina Econòmica
- Dia: 1
- Demanda de Potència: 250.0
- Potència Generada: 225.0
- Demanda de Potència Satisfeta: 90.0 %
- Beneficis: 225.0 Unitats Econòmiques
- Penalització Excés Producció: 0.0 Unitats Econòmiques
- Cost Operatiu: 215.0 Unitats Econòmiques
- Guanys acumulats: 10.0 Unitats Econòmiques
```

Classe `PaginaEstat`

La classe `PaginaEstat` hereta de `PaginaBitacola` i ha de tenir el atributs necessaris segons el que es va a explicar a l'Apartat 2.4.2. El atributs s'han d'inicialitzar usant

[3] l'inserció de les barres seria del 90%, el reactor estaria activat i també una bomba refrigerant.

el constructor. La demanda de potència satisfeta es pot calcular a partir de la resta d'atributs. Addicionalment, s'hauran de definir *getters* per a cada atribut.

A més, la classe `PaginaEstat` hauria de tenir un mètode `toString` que generi una representació textual de la pàgina. Un exemple de la cadena a generar seria[4]:

```
# Pàgina Estat
- Dia: 1
- Inserció Barres: 90.0 %
- Output Reactor: 125.0 Graus
- Output Sistema de Refrigeració: 125.0 Graus
- Output Generador de Vapor: 112.5 Graus
- Output Turbina: 225.0 Unitats de Potència
```

Classe `PaginaIncidencies`

La classe `PaginaIncidencies` hereta de `PaginaBitacola` i conté una llista de `String` descrivint incidències, segons es va explicar a l'Apartat 2.4.3.

La classe ha de definir el mètode `afegeixIncidencia`, que afegeix una incidència a la llista d'incidències per a un dia concret. El mètode ha de tenir el següent prototip:

```
public void afegeixIncidencia(String descIncidencia);
```

Per a aquesta classe també s'ha definir el mètode `toString` que generi cadenes semblants a la següent:

```
# Pàgina Incidències
- Dia: 4
- Descripció Incidència: El reactor es va desactivar per superar
la temperatura màxima
- Descripció Incidència: La bomba refrig. 2 està fora de servei
```

Classe `Bitacola`

La classe `Bitacola` ha de definir un atribut de tipus `ArrayList` per guardar objectes de tipus `PaginaBitacola`. A més, la classe ha d'implementar la interfície `InBitacola` i definir el mètode `toString` que generi una representació com a `String` de totes les pàgines de la bitàcola separades per salts de línia (`'\n'`).

Classe `Dades`

La classe `Dades` és la classe principal del paquet del model, atès que conté i gestiona totes les dades de l'aplicació.

Per establir amb claredat el que ha de fer aquesta classe, s'ha creat la interfície `InDades`, que proveeix una llista amb els mètodes a implementar.

[4] de nou, l'inserció de les barres seria del 90%, el reactor estaria activat i també una bomba refrigerant.

Els mètodes proveïts per `Dades` seran utilitzades per la classe `Adaptador`. No hauria de ser necessari crear mètodes addicionals dins de Dades a més del constructor, els mètodes de `InDades` i els mètodes de servei declarats al codi base: `refrigeraReactor`, `revisaComponents` i `actualitzaEconomia`.

A continuació es proporcionen detalls addicionals sobre alguns mètodes:

- **Constructor**: aquest mètode es proporciona completament implementat al codi base i no es pot modificar. **Inicialment, la turbina i el generador de vapor estan activats i aquest estat mai es modificarà. En canvi, les bombes refrigerants i el reactor estan inicialment desactivats**. Es recomana estudiar el constructor en profunditat com a ajuda per a definir els atributs de la classe.

- **Mètode `mostraEstat`**: genera una pàgina de bitàcola d'estat mostrant la configuració actual de la central.

- **Mètode `calculaPotencia`**: aquest mètode ha de retornar la potència generada per la central, que es correspondria amb l'output de la turbina en unitats de potència segons la configuració actual de la central. Per a implementar aquest mètode, s'hauria d'evitar replicar el codi utilitzat dins de `mostraEstat`.

- **Mètode `finalitzaDia`**: aquest mètode es proporciona implementat al codi base i no pot modificar-se. El mètode s'encarrega d'executar els pasos necessaris per finalitzar el dia. Amb aquest propòsit, `finalitzaDia` utilitza els mètodes de servei `actualitzaEconomia`, `refrigeraReactor` i `revisaComponents` (el codi d'aquests mètodes s'ha de completar). El mètode retorna un objecte de tipus `Bitacola` amb tres pàgines que descriuen la situació actual de la central, incloent la seva economia, estat i incidències.

- **Mètode `actualitzaEconomia`**: aquest mètode ha d'implementar les acciones per actualitzar l'economia de la central descrites a l'Apartat 2.3.1. Per això es farà servir el mètode `calculaPotencia`, que obté la potència generada per la central segons la seva configuració actual.

- **Mètode `getGuanysAcumulats`**: aquest mètode és només un *getter* per a l'atribut `guanysAcumulats` de la classe `Dades`.

- **Mètodes `refrigeraReactor` i `revisaComponents`**: aquests mètodes han de dur a terme les accions explicades a l'Apartat 2.3.2.

Classe `Adaptador`

La classe `Adaptador`, com ja s'ha explicat anteriorment, serà utilitzada per intervenir entre la vista i les dades del model. Per això, la classe `Adaptador` farà servir un atribut de la classe `Dades`, per ser la classe principal del nostre model.

Un exemple de servei proporcionat per la classe `Adaptador` seria convertir la llista d'incidències retornada pel mètode `mostraIncidencies` de la classe `Dades` com a un únic objecte de tipus `String` on les incidències es mostren separades per salts de línia (`'\n'`). Això permet que la vista funcioni sense tenir coneixement dels detalls d'implementació del model.

Tal com es veia a l'Apartat 3, la nostra aplicació ha de ser capaç de guardar i carregar les dades del model a partir de fitxers (persistència de les dades). Es suggereix que aquesta tasca s'implementi mitjançant dos mètodes de la classe `Adaptador`, amb prototips:

```
public void guardaDades(String camiDesti) throws CentralUBException;
public void carregaDades(String camiOrigen)throws CentralUBException;
```

Perquè les dades del model puguin carregar-se i guardar-se correctament, es recorda que les classes involucrades han d'implementar la interfície `Serializable`.

Si es necessita informació addicional sobre el tractament de la persistència de les dades en Java, es pot consultar el material de suport per a la pràctica anterior (Pràctica 2) disponible al Campus Virtual.

Classe `CentralUB`

Defineix el menú d'opcions i la lògica de l'aplicació. Per a això, definirem una funció anomenada `gestioCentralUB` amb la següent signatura:

```
public void gestioCentralUB();
```

Aquesta funció internament utilitzarà la classe `Menu` que es proporciona com a material de la pràctica.

El codi base inclou un fitxer amb una plantilla per a la classe `CentralUB`, que incorpora el que es necessita per generar la demanda de potència de cada nou dia i un mètode de servei per a finalitzar el dia en curs.

És molt important ressaltar que des de `CentralUB` no es pot importar cap classe del paquet model. Només podem importar la classe `Adaptador`.

Classe `IniciadorCentralUB`

Conté la classe principal de l'aplicació i està ubicada al paquet `prog2.vista`. Declara un objecte de la classe `CentralUB` i crida a mètode `gestioCentralUB` esmentat anteriorment.

Classe `CentralUBException`

S'ha de crear la classe `CentralUBException` dins del paquet `prog2.vista`. Aquesta classe serà utilitzada per gestionar excepcions, tal com s'explica més avall.

6 Excepcions

Igual que en la pràctica anterior, la gestió dels errors es durarà a terme mitjançant excepcions. **Les operacions que poden causar excepcions es resumeixen a la Taula 1.**

El primer pas per implementar el maneig d'excepcions és crear la classe `CentralUBException`. Un cop creada (tal com vam veure en la pràctica anterior), hem de fer servir `throw` des dels mètodes en què es pugui produir algun error. Farem servir la paraula clau `throws` al final de la capçalera del mètode en cas que

vulguem delegar la captura d'una excepció fins al mètode on s'hagi de gestionar. El constructor de l'excepció ha de servir per introduir el missatge descrivint l'error. Finalment, utilitzarem un bloc `try-catch` per capturar l'excepció i informar l'usuari de l'error produït i controlar el flux d'execució. No s'ha de fer `try-catch` amb `print` fora de les classes de la vista.

Es recorda que es pot consultar informació addicional sobre el maneig d'excepcions en el material de suport per a la Pràctica 1 disponible al Campus Virtual.

7 Classes de test

Per comprovar el correcte funcionament del codi implementat és necessari córrer un conjunt de casos de prova (Unit Tests) que heu de definir en diverses classes de test corresponents a cadascuna de les classes del model i que hauran d'estar creades a la carpeta test. Un cas de prova (Unit Test Case) és una part de codi, que assegura que una altra part del codi (un mètode) funciona com s'esperava. Un cas de prova es caracteritza per una entrada coneguda i un resultat esperat. JUnit és un marc de proves unitàries per al llenguatge de programació Java (https://junit.org/junit5/,https://www.jetbrains.com/help/idea/junit.html). Teniu un document de suport sobre JUnit al Campus Virtual.

8 Format del Lliurament

La Pràctica 3 es donarà per acabada quan s'acabin les dues parts que la componen i es corregirà llavors. No obstant això, cal fer el lliurament de la Pràctica 3 Part 1 junt amb una memòria que contindrà només dos diagrames.

Es recomana complir la data de lliurament de la tasca oberta en el campus virtual per poder començar el treball de la part 2 a continuació. El treball de la part 2 consistirà en el desenvolupament d'un altre projecte de IntelliJ que està descrit a un enunciat separat corresponent a la part 2 de la pràctica.

Pel lliurament de la Pràctica 3 Part 1 s'ha de generar un fitxer comprimit (ZIP) amb el nom dels membres de la parella: `Cognom1Nom1Cognom2Nom2_P31`, que contingui:

1. El projecte de IntelliJ: Tot el codi generat ha d'estar correctament comentat per poder executar el javadoc, generant automàticament la documentació en línia del codi.

2. Una memòria amb:

   - El diagrama de relacions entre les classes que has utilitzat a la pràctica 3 part 1. No cal incloure la llista d'atributs i mètodes.
   - El diagrama de seqüencia per mostrar la interacció entre objectes quan s'executa l'opció de mostrar l'estat de la central.

9 Data Límit del Lliurament

Consulteu la data límit de la tasca oberta al Campus Virtual.