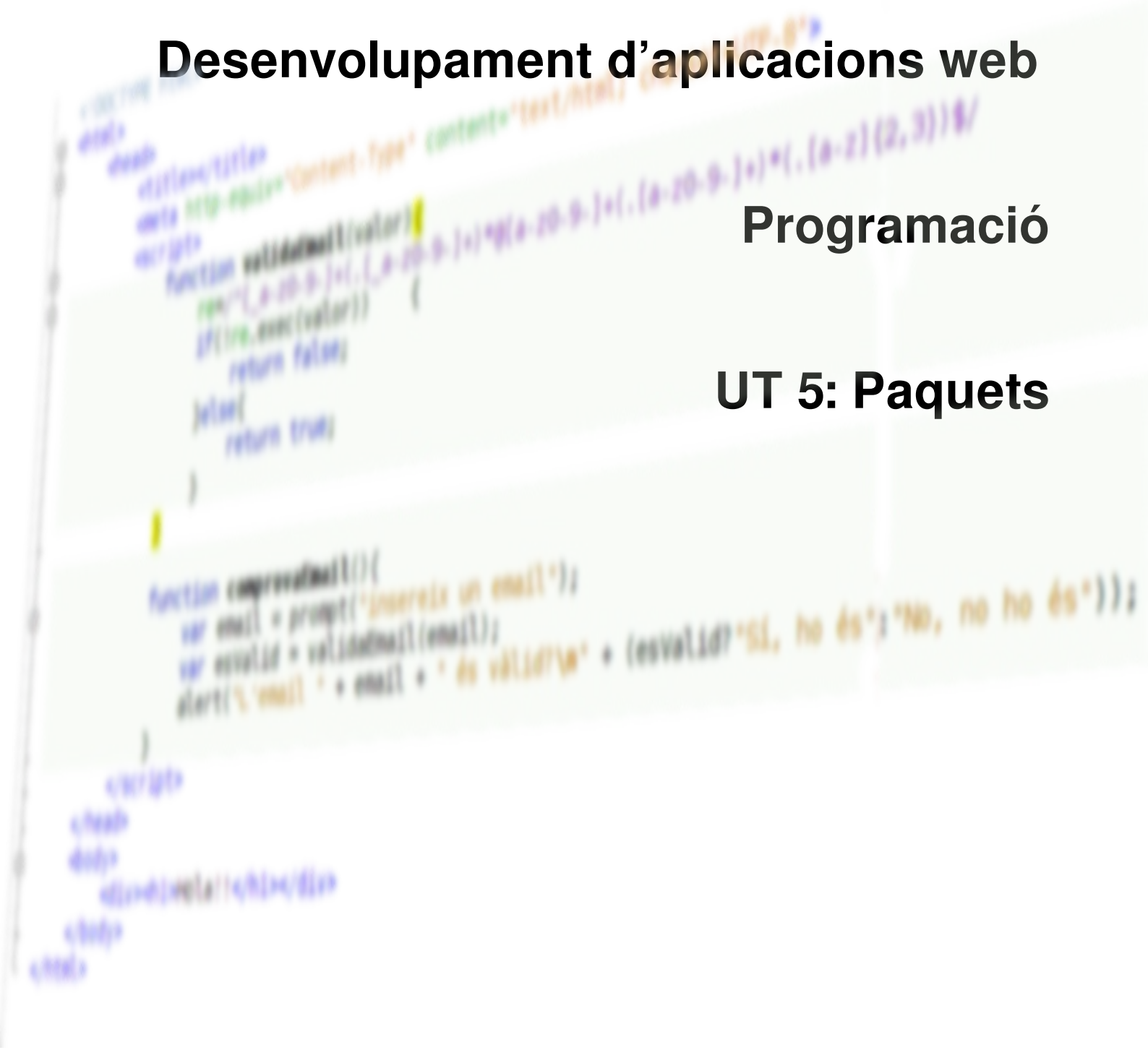
PAU CASESNOVES
Centre Integrat de Formació Professional

# Desenvolupament d'aplicacions web

## Programació

## UT 5: Paquets





Joan Pons Tugores

# Índex de continguts

# Paquet

Un paquet, *package*, és una agrupació de tipus (classes, enumeracions, ...) relacionats que proporciona protecció d'accés i un espai de noms propi.

- **Protecció d'accés**: El modificador d'accés protected o no especificar-ne cap permet l'accés a objectes d'altres classes del mateix paquet i l'impedeix a classes d'altres paquets.
- **Espai de noms propi**: Podem tenir classes distintes amb el mateix nom a diferents paquets. En realitat, el nom complet de la classe està format per nomPaquet.nomClasse, per tant podem tenir paquet1.Classe i paquet2.Classe

**Tots els tipus que definiu, classes, enumeracions, ... han de pertànyer a un paquet.**

## Raons per utilitzar paquets

Es convenient agrupar classes en un paquet per distintes raons:

- Ajuda als desenvolupadors a veure quines classes estan relacionades.
- Ajuda als desenvolupadors a localitzar fàcilment una determinada classe.
- Ens assegura que els noms de les nostres classes no provocaran conflictes amb altres classes perquè el paquet crea el seu propi espai de noms.
- Ens permet que els tipus del paquet tenguin accés complet entre ells i restringir-ne l'accés a altres tipus de fora del paquet.

## Creació d'un paquet

Per crear un paquet basta elegir-ne el nom i posar la sentència

```
package nomPaquet;
```

a l'inici de cada fitxer font que inclogui el codi de les classes i enumeracions que volgueu incloure al paquet. Aquesta instrucció ha de ser la primera del fitxer font i ha de ser única, és a dir, un tipus només pot pertànyer a un paquet.

## Nomenclatura

Hem dit que el nom complet d'una classe està format pel nom del paquet i el de la classe. Això ens permet definir classes amb el mateix nom a distints paquets. Com evitam repetir però el nom del paquet amb milions de programadors picant codi? **La convenció és utilitzar el domini de l'empresa al revés**. Per exemple, si el domini de l'empresa és *programadors.com* els nostres paquets haurien de començar per *com.programadors,* per exemple *com.programadors.dades*

Dins cada empresa és problema seu com ho facin per evitar col·lisions. Una tècnica habitual és afegir-hi el nom del projecte, per exemple *com.programadors.banc.dades.*

Les regles de nomenclatura diuen que el nom del paquet ha d'estar format només per minúscules

```
com.programadors.dadesbanc;
```

i no

```
com.programadors.dadesBanc;
```

## Utilització dels membres d'un paquet

Els membres d'un paquet són els tipus que el paquet agrupa. Per utilitzar-los fora del paquet s'ha de fer d'una de les tres maneres següents:

- Utilitzar el nom complet del tipus, és a dir *nompaquet.NomTipus*. Si s'ha de repetir diverses vegades, apart de que sigui pesat d'escriure, dificulta la lectura i comprensió del codi.
- Importar el tipus. Això ens permetrà utilitzar només el nom del tipus sense haver d'especificar cada vegada el paquet. Els *imports* han d'anar just davall la declaració de paquet. Es pot utilitzar quan utilitzam pocs tipus del paquet.

```
import com.programadors.banc.Client;
```

- Importar el paquet sencer. Es fa posant un asterisc al darrera del nom del paquet. Així importam tots els tipus del paquet. Útil quan hem d'importar la majoria de tipus.

```
import com.programadors.banc.*;
```

El compilador automàticament importa els següents paquets: *java.lang* i el paquet del que forma part el tipus.

## Jerarquia de paquets

Encara que ho pugui semblar els paquets no formen cap jerarquia. Per exemple, si tenim els següents paquets:

```
com.programadors.banc;

com.programadors.banc.dades;

com.programadors.banc.gestor;
```

En realitat són tres paquets completament independents. Per tant, fer un import de *com.programadors.banc.** no inclou els tipus que hi pugui haver a *com.programadors.banc.dades* o *com.programadors.banc.gestor*, sinó només els que hi pugui haver dins *com.programadors.banc*.

## Ambigüitats

Pot ser que importem dues classes amb el mateix nom de dos paquets diferents. En tal cas, no ens quedarà més remei que utilitzar el nom complet de cada classe.

## Fitxers java i class

L'estructuració en paquets de les classes Java no implica cap jerarquia de fitxers. Podem tenir totes les classes juntes dins el mateix directori. Així i tot, és una pràctica habitual crear una jerarquia de directoris que replica el nom del paquet.

Es sol crear un directori per a cada paquet. Per exemple, per *com.programadors.banc* tendrem un directori *com*, amb un subdirectori *programadors*, amb un subdirectori *banc*.

Normalment tendrem aquesta estructura per els fitxers font, .java, i la mateixa estructura repetida per els fitxers compilats, .class. **No s'aconsella tenir els fitxers de codi font als mateixos directoris que els fitxers compilats**.

# Distribució: jar

Per que es pugui utilitzar una classe Java és suficient amb distribuir aquesta classe. Des de qualsevol ordinador amb una màquina virtual instal·lada és podrà executar. Rarament una aplicació Java estarà formada per una única classe. Llavors resulta *engorrós* haver de generar un zip, desempaquetar-lo al client, ...

Per facilitar la tasca de distribuir i reutilitzar codi en altres aplicacions disposam dels fitxers *.jar* (***j***ava ***ar***chive).

Aquests fitxers ens permeten agrupar tot el codi de la nostra aplicació en un únic fitxer, comprimit o no, i executar-lo directament, sense haver de manipular-lo. A més ens ofereixen seguretat, es poden signar digitalment, faciliten el control de versions, la portabilitat,..

## Execució d'una aplicació empaquetada amb jar

```
java -jar jar-file
```

Executa l'aplicació que s'ha empaquetat en el fitxer .jar

Com sabem quina classe és l'inici de l'aplicació?

## Manifest

El manifest és un fitxer especial de text que conté informació sobre el fitxer jar i el seu contingut. En crear el jar es crea automàticament un manifest. Només pot haver-ni un a cada jar i la seva ruta i nom sempre són els mateixos:

```
META-INF/MANIFEST.MF
```

Està format per parelles etiqueta valor separades per :

Al manifest podem establir el punt d'entrada a l'aplicació, la classe que s'executarà per a iniciar-la. Hem d'afegir una línia al manifest com aquesta:

```
Main-Class: classname
```

El nom de la classe ha d'incloure el *package*, i evidentment, ha de ser una classe que tengui un mètode *public static void main(String[] args)*

# Utilització de jar als IDE

## Eclipse

- Per utilitzar classes d'un altre jar:

  ```
  Propietats del projecte -> Java Build Path -> Libraries -> Add Jars
  ```

- Per a construir el jar del nostre projecte quan es tracta d'una llibreria

  ```
  File -> Export -> Java -> Jar File
  ```

  L'assistent ens anirà demanant dades per crear el manifest. També podem crear un fitxer de text amb el manifest i incloure'l.

- Per a construir el jar del nostre projecte quan es tracta d'una aplicació

  ```
  File -> Export -> Java -> Runnable jar File
  ```

  L'assistent ens demanarà quina configuració d'execució volem utilitzar i com volem tractar les llibreries necessàries.

## Netbeans

- Per utilitzar classes d'un altre jar:

  Propietats del projecte -> Libraries -> Add Jars

- Per a construir el jar del nostre projecte

  Propietats del projecte -> Construcció -> Empaquetament

  L'assistent ens demana on volem crear el fitxer, quins fitxers volem excloure, si el volem comprimit o no, ...

  Per defecte en haver compilat el projecte crea el jar. El podem trobar a la carpeta dist.

  Si hem especificat una classe principal a Propietats del projecte -> Execució la inclourà al manifest del jar com a punt d'entrada a l'aplicació.

## intelliJ

- Per utilitzar classes d'un altre jar:

  File → Project structure → Libraries → + → Java i localitzar el jar al sistema de fitxers

- Per a construir el jar del nostre projecte
  - Primer hem de definir el jar com un artifact.

    File → Project structure → Artifacts → + → JAR → From modules with dependencies ...

    Triau el mòdul i la classe principal si n'hi ha. Les altres opcions poden quedar amb els valors per defecte.

  - En voler generar el jar:

    Buid → Build artifacts i seleccionar el jar i l'opció que volem executar, per exemple Build.

    El jar es genera dins la carpeta *out → artifacts.*

## Visual Studio Code

- Per utilitzar classes d'un altre jar. Hem de tenir oberta la carpeta, no només un fitxer. Llavors:

  > Java projects → Referenced Libraries → + → Java i localitzar el jar al sistema de fitxers

- Per a construir el jar del nostre projecte

  > Java projects → Referenced Libraries → Pitjar la icona d'una fletxa després del +

  Haureu d'especificar la classe principal si n'hi ha, que voleu incloure al jar, el bin amb els .class i els jar de les llibreries.