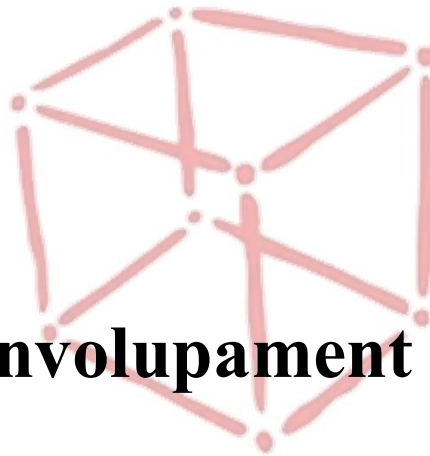


**Entorns de desenvolupament.**



**Tema 4:**



**Entorns de desenvolupament integrat.**



# ENTORN DE DESENVOLUPAMENT

Ja hem vist com és el procés de desenvolupament de codi: el programador escriu el codi font, corregeix les errades, el compila, l'enllaça a les llibreries i obté el codi executable.

Totes les eines que utilitzi per a realitzar aquestes tasques formen l'entorn de desenvolupament utilitzat per aquest desenvolupament.

Eines que poden formar part d'aquest entorn:

- Un editor de text: notepad, notepad++, gedit, ...
- El compilador
- L'enllaçador
- Un debugger

- Màquines virtuals
- Intèrprets
- ...

Per exemple, per desenvolupar una aplicació java necessitem només un editor de text, el compilador *javac* i el JRE, amb *java* per executar el programa i fer proves.

Cada vegada que volem comprovar uns canvis que haguem al codi haurem de

- Guardar el codi font.
- Anar a la finestra de comandes o terminal i allà
- Compilar el codi font amb *javac*
- Si hi ha errors,
  - mirar a quina línia ens diu el compilador que es troba l'error
  - Tornar a l'editor de text
  - Cercar la línia
  - Corregir-lo
- Si no hi ha errors,
  - executar-lo

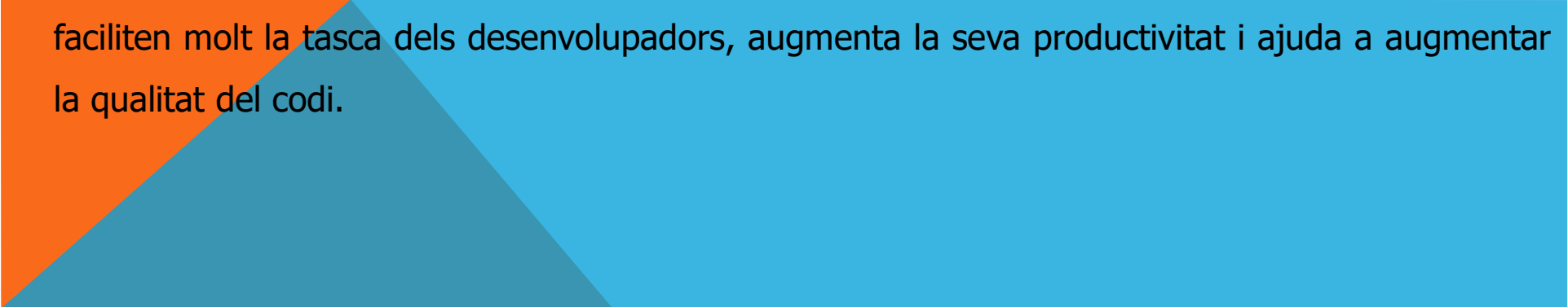
# ENTORN DE DESENVOLUPAMENT INTEGRAT

Un entorn de desenvolupament integrat, a partir d'ara IDE, és un programari que inclou totes les eines necessàries per a desenvolupar codi en un llenguatge determinat o inclús en diversos llenguatges.

La idea de reunir en una sola aplicació totes les eines necessàries per al desenvolupament de programari cerca facilitar la tasca del programador, millorar el seu rendiment i augmentar la qualitat del codi elaborat.

Per exemple en un IDE, com ara Netbeans, compilar i executar un programa és tan fàcil com pitjar un botó que fins i tot ens guarda el codi font si hi havia modificacions abans de compilar-lo i executar el resultat final.

L'elecció d'un IDE adequat al projecte que estem desenvolupant i la seva utilització eficient faciliten molt la tasca dels desenvolupadors, augmenta la seva productivitat i ajuda a augmentar la qualitat del codi.



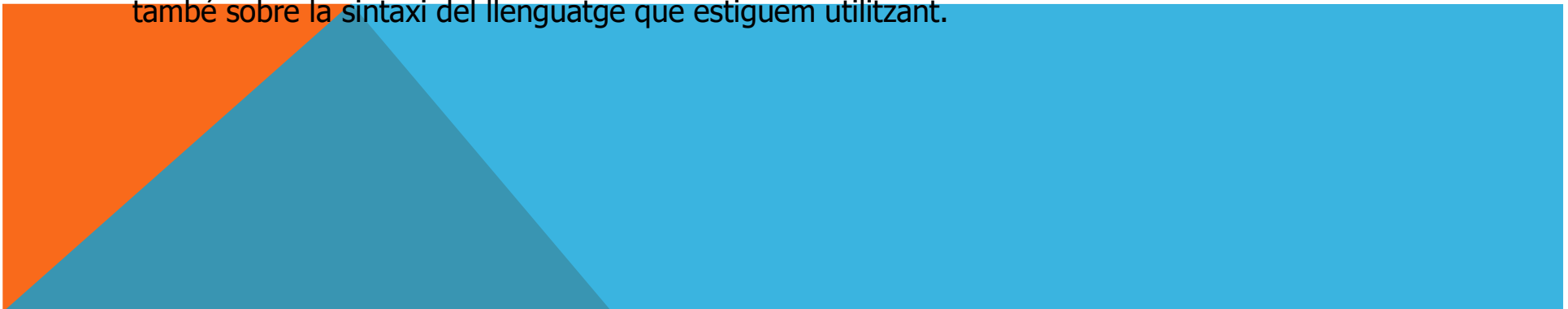
# FUNCIONS D'UN IDE

- **GUI: Interfície gràfica d'usuari.** Normalment un IDE disposa d'una interfície gràfica amb botons, finestres, ... que reuneix en un sol lloc totes les possibilitats de l'entorn.
- **Editor de text:** Es impensable un entorn dedicat a picar codi sense un editor de text que ens ho permeti. Els editors solen incorporar funcions avançades de les que no disposa el bloc de notes, com per exemple:
  - **Resaltat de la sintaxi:** Mentre s'escrivim, en reconèixer una paraula clau del llenguatge, o un literal els escriu amb una tipografia diferent per facilitar la lectura del codi.
  - **Detecció d'errades i advertències:** Mentre escrivim, analitza el codi i si detecta errades de compilació marca la línia on es troben.
  - **Compleció codi:** Ens ajuda a completar el codi. En escriure una el nom d'una classe o d'un objecte ens mostra els seus membres, ...
  - **Regions desplegable:** Permet definir apartats de codi que es poden amagar o desplegar a voluntat per facilitar la visualització del que realment ens interessa.
  - ...

- **Compilador:** Si l'IDE està pensat per treballar amb un llenguatge compilat normalment incorporarà un compilador per a generar el codi executable. Normalment localitzarà les errades i ens dirà on són. Si no n'hi ha generarà el codi executable.
- **Intèrpret:** Si l'IDE està pensat per treballar amb un llenguatge interpretat normalment incorporarà un intèrpret o qualche mecanisme per llençar un intèrpret i provar el codi.
- **Depurador:** En anglès debugger. És un programa que permet provar i trobar errades al nostre codi en temps d'execució. Algunes de les seves funcions típiques són:
  - Punts d'interrupció: Aturar l'execució del codi en una certa instrucció del programa.
  - Control de variables: Veure en tot moment el valor que van prenent les variables, normalment és útil en combinació amb els punts d'interrupció.
  - Execució pas a pas del codi: Per veure per on va passant, quines instruccions s'executen i quins valor prenen les variables.



- **Accés a bases de dades:** Opcionalment poden incloure eines d'accés a bases de dades per controlar la interacció amb elles del nostre programa.
- **Gestió d'arxius:** Normalment incorporen un gestor d'arxius que ens permet moure fitxers dins el projecte, copiar-los, eliminar-los...
- **Control de versions:** Solen incorporar qualche sistema de control de versions, molt útil sobre tot en treballar en equip.
- **Refactorització:** És una tècnica, o millor dit, tota una sèrie de tècniques, per canviar l'estructura del codi sense alterar la seva funcionalitat però millorant la seva eficiència. Els IDEs solen incorporar eines per a fer-ho.
- **Documentació:** També solen incorporar eines que faciliten la documentació de l'aplicació, per exemple JavaDoc.
- **Ajuda:** La gran majoria incorpora diferents eines d'ajuda, no només sobre la utilització de l'IDE, sinó també sobre la sintaxi del llenguatge que estiguem utilitzant.



# INSTAL·LACIÓ D'UN IDE

La instal·lació d'un entorn de desenvolupament integrat, com podeu suposar no segueix un patró general, cada un té les seves peculiaritats. Si volem treballar en Java el primer que haurem de fer és instal·lar el JDK. El que ve després ja depèn de l'IDE en qüestió.

## Netbeans

La instal·lació Netbeans es realment senzilla.

- Baixem de la seva pàgina la versió que ens interressi tenint en compte perquè el volem utilitzar(quines utilitats inclourà) i l'arquitectura del sistema en el que l'utilitzarem <https://netbeans.org/downloads/>
- Descomprimim l'arxiu en un directori sobre el que tinguem permisos d'execució. Intenteu no fer-ho a l'escriptori. Es crearà una carpeta *Netbeans*.
- Executem Netbeans de dins aquesta carpeta.
- Ens demanarà quin workspace volem utilitzar. Podríem dir que el workspace és el magatzem on tindrem les nostres aplicacions. Escollim una carpeta i si marquem la casella no ens tornarà a fer aquesta pregunta.



# UTILITZACIÓ D'UN IDE

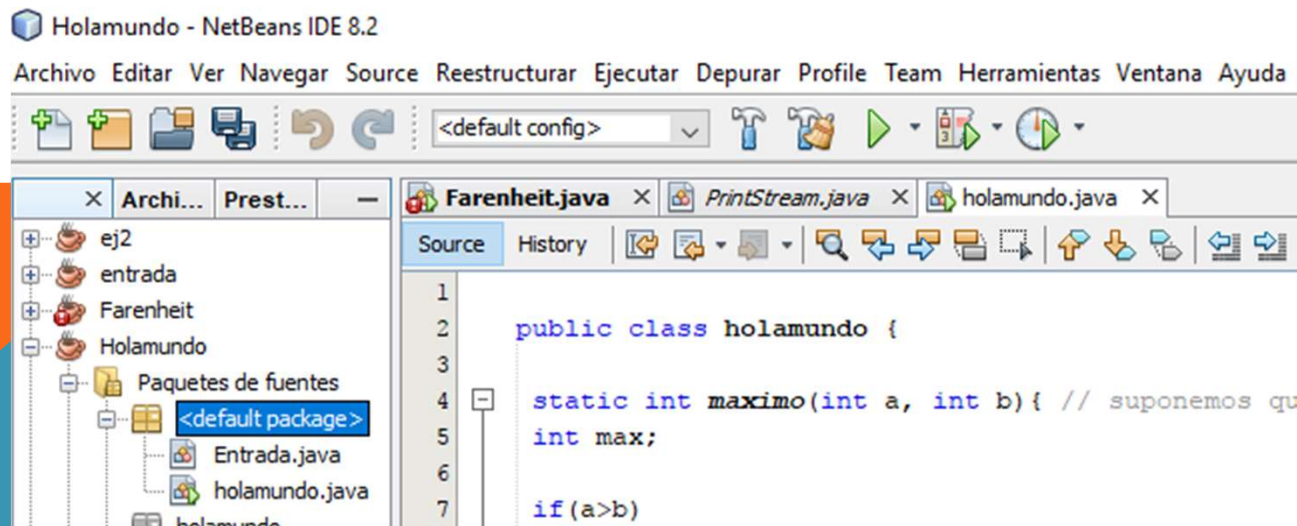
## Projecte

Els IDE's de Java normalment utilitzen el concepte de projecte. Un **projecte** és tot el conjunt de fitxers, tant de codi com auxiliars (imatges, text, ...), paquets, ... que formen part d'una aplicació.

Normalment els projectes només tenen una classe executable, una classe que tingui el mètode *main*. Nosaltres, per començar ens botarem aquesta limitació i farem que cada exercici tingui la seva pròpia classe dins el *main*.

El primer que farem serà crear un projecte Java. Anem al menú *File->NewProject*.

Al diàleg que s'obrirà posarem el nom del projecte i pitjarem el botó *Finish*.



A la imatge anterior podeu veure l'aspecte de la perspectiva de java de Netbeans:

- A l'esquerra teniu l'explorador de projectes. Hi teniu tots els projectes del vostre *workspace*. Si desplega un projecte apareix una carpeta *src* on tindreu tot el vostre codi font, tots els vostres arxius *.java*
- Des d'aquesta finestra podeu copiar, renombrar(refactor), eliminar arxius. És aconsellable fer-ho sempre des d'aquí i no directament des del sistema operatiu. Per exemple, si copieu un arxiu Java vos demanarà el nom de la còpia i posarà el mateix nom a la classe dins el codi. Recordeu que la classe i el fitxer han de tenir exactament el mateix nom.
- Al centre hi trobeu l'editor de text. Podeu veure que pinta les paraules reservades Java d'un color, les cadenes de text d'un altre, ...
- També ens marca a quina línia hi ha errades de sintaxi, amb una creu vermella a l'esquerra (si posem el cursor al damunt ens diu quin és l'error, i si el pitja'm ens proposa una solució) i amb una línia vermella a l'esquerra que és útil en programes llargs ja que clicant-hi a sobre ens porta a la línia en qüestió.
- Si vos apareix una icona d'una bombolla groga i una exclamació significa que en aquella línia hi ha un warning (una alerta), no és una errada però s'hauria de revisar
- Per mostrar els nombres de línia(molt recomanable) heu de pitjar el botó dret del ratolí sobre la barra on apareix la creu vermella i marcar *Show line numbers*.

- A la dreta hi trobareu
  - Una finestra de tasques, que es pot utilitzar per posar recordatoris de coses pendents
  - Una finestra que posa outline i que en programes complexes és molt útil perquè permet accedir directament a mètodes o atributs de la classe que estam editant.
- Finalment a la part inferior teniu una finestra amb una sèrie de pestanyes:
  - Consola: simula la “finestra negra” del windows. Apareixen els missatges que volem mostrar i hi escriurem per teclat el que ens demani el programa.
  - Problems: Mostra les errades i les alertes del projecte actual. Si clicam a sobre d'un d'ells ens porta a la línia que provoca l'errada.
  - Javadoc: Ens mostra l'ajuda de les classes de l'API de Java.



A part d'aquestes finestres, la perspectiva es pot modificar des del menú *Windows*:

- *Customize perspective* permet modificar els botons de la barra d'eines, comandes dels menús, ...
- *Show view*: Mostrar plafons de la perspectiva, molt útil si sense voler en tanca'm qualche un.
- *Preferences*: Ens permet, entre moltes altres opcions, canviar el tipus de lletra i el color de l'editor, o com es ressalten les paraules clau, els comentaris, ...
- ...

Una particularitat molt útil és que Netbeans ens permet veure el codi de més d'un fitxer a la vegada. Per fer-ho hem d'arrossegar la pestanya del fitxer en qüestió i arrossegar-la per la pantalla. Apareixeran unes línies que ens indiquen com quedarà configurada. En tenir-ho com volem, amollem el fitxer i quedarà configurada d'aquesta manera. Fins i tot es pot treure un fitxer de la finestra de windows, per exemple per posar-lo a la pantalla auxiliar.

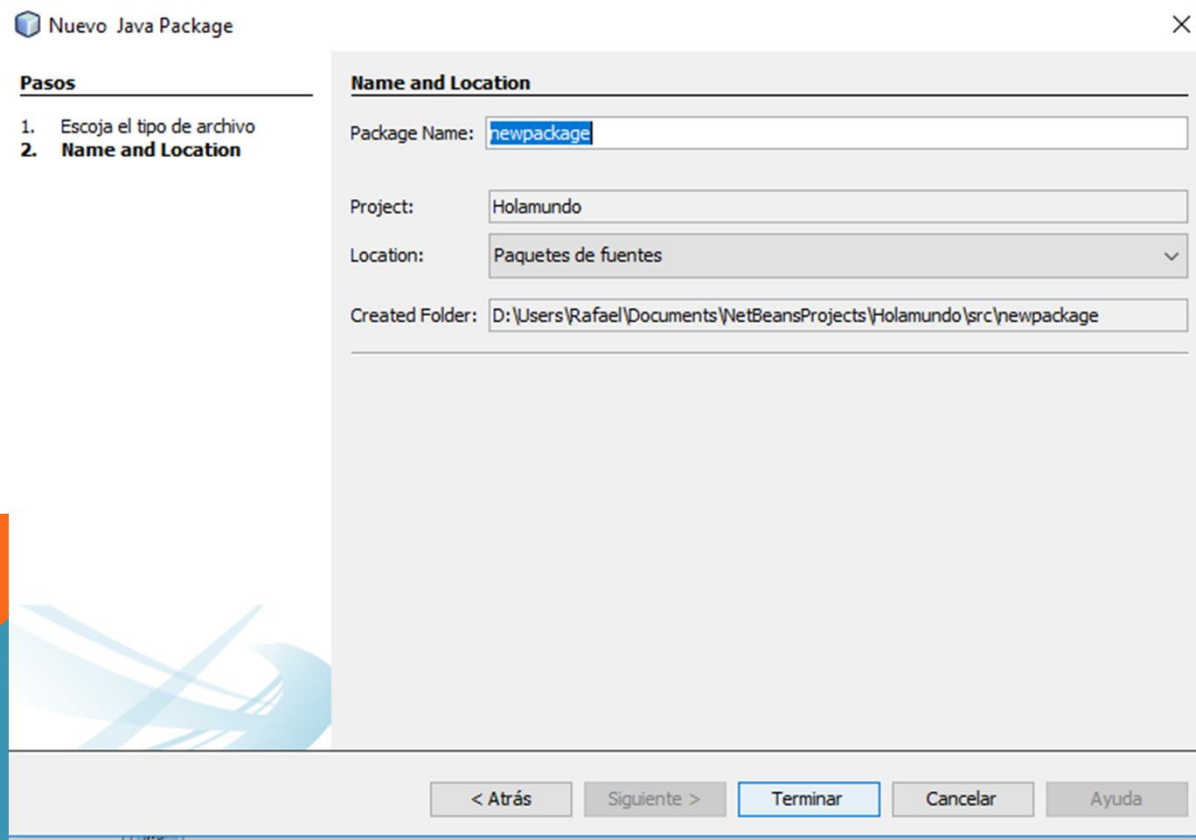


## Package

Els paquets, en projectes grossos, permeten ordenar les distintes classes que formen el projecte. Segons l'estàndard de Java totes les classes haurien d'estar dins un paquet. Per això en crearem un dins el nostre projecte.

Ens situem a sobre del projecte a l'explorador i pitja'm el botó dret *New->Package*.

Posem el nom del paquet, en minúscules, i pitja'm *Finish*.



## Package

Un **paquet** és una **estructura lògica**, no física. **Una classe pertany a un paquet perquè s'especifica a la seva declaració, no perquè estigui dins una carpeta o una altra.**

Així i tot la pràctica habitual, i el que fan per defecte la majoria d'entorns de desenvolupament, és crear una estructura de carpetes on es guarden els fitxers de les classes que segueix l'estructura dels paquets.

A més d'organitzar les classes, els paquets permeten definir l'accés que permetrà un objecte als seus membres. **Podem definir que un membre sigui accessible només per objectes de la mateixa classe, de classes del mateix paquet o per qualsevol altre objecte.**



## Classe

Qualsevol codi que vulguem programar en Java haurà d'estar dins una classe. Per crear una classe ens situem a sobre del package que la contindrà i pitja'm el botó dret *New->Class*.

El nom de la classe sempre ha de començar amb majúscula. Vegeu l'inici del capítol de sintaxi. Pitja'm el botó *Finish* i tindrem la classe creada.

Nuevo Java Class

**Pasos**

1. Escoja el tipo de archivo
2. **Name and Location**

**Name and Location**

Class Name:

Project:

Location:

Package:

Created File:

Warning: It is highly recommended that you do not place Java classes in the default package.

< Atrás    Siguiente >    **Terminar**    Cancelar    Ayuda

## Disponibilitat de les classes

Cadascuna de les classes definides en el projecte podrem accedir als seus mètodes depenent de la definició dels seus modificadors d'accés i de a on es troben en l'estructura de paquets dels projectes.

**En general podríem dir que tots els atributs haurien de ser privats i crear mètodes públics que permetin accedir-hi**, així controlem millor el comportament de l'objecte.

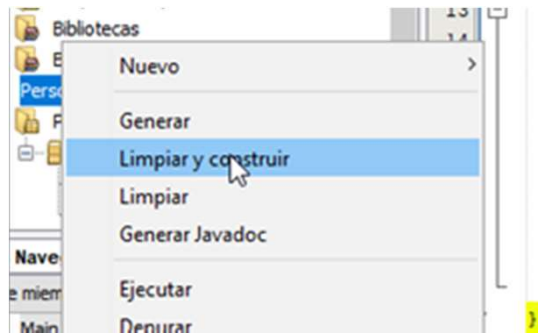
Visibilitat	Public	Protected	Default	Private
Mateixa classe	SI	SI	SI	SI
Qualsevol classe i mateix paquet	SI	SI	SI	NO
Subclasse i mateix paquet	SI	SI	SI	NO
Subclasse i fora paquet	SI	SI, PER HERENCIA	NO	NO
Qualsevol classe fora paquet	SI	NO	NO	NO



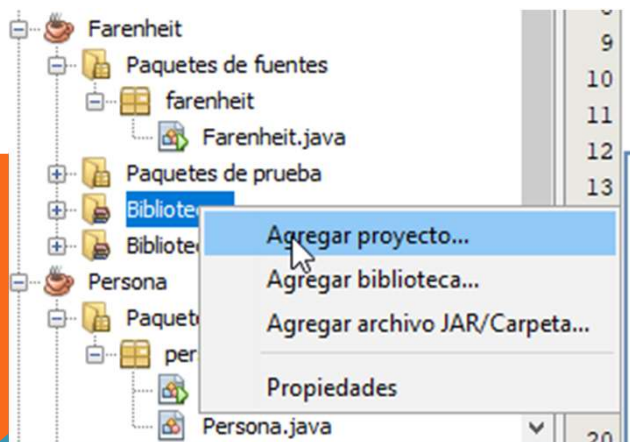
## Disponibilitat de les classes II

A més, es pot donar la situació que necessitem una classe que es troba en altre projecte. Per poder comptar amb ella, n'hi que importar aquesta classe com a part de la nostra biblioteca del projecte. Ho farem així:

1- Generem un .jar del projecte on es troba la classe que volem importar. Botó dret damunt del projecte→limpiar y contruir.



2- Importem a la nostra biblioteca Generem un .jar del projecte on es troba la classe que volem importar. Botó dret damunt del projecte→limpiar y contruir.



## Execució

Amb Netbeans no fa falta compilar explícitament el codi font. Cada vegada que es guarda el fitxer es compila.

Per executar una classe només ens hem de situar dins el codi de la classe i pitjar el botó verd amb una fletxa blanca dins de la barra d'eines Netbeans. Això compilarà i executarà la classe.

*Netbeans* permet definir configuracions d'execució, per poder executar el projecte en distintes condicions. De moment no ens farà falta utilitzar-les. Algunes de les parametritzacions que ens permet:

- Elegir la classe que s'executarà en iniciar el projecte.
- Amb quina màquina virtual s'executarà si n'hi ha més d'una instal·lada
- Quins arguments es passen a l'aplicació des de la línia de comandes,
- ...



## Connectors

Per si no ens basta tot el que duu Netbeans es pot ampliar la seva funcionalitat afegint connectors, en anglès *pluggins*. Podem instal·lar els que vulguem, però hem de tenir en compte que la instal·lació de connectors penalitza el rendiment de l'aplicació, sobre tot augmenta el temps d'arrencada.

