Présentation d'IntelliJ IDEA

Memorize build-3.49.727

Synthèse de la version 0.6.15 du cours.

Quentin 'Calistro' RAMSAMY- -AGEORGES

March 25, 2021

https://memorize.lgs-games.com/fr/latest/utils/jetbrains.html

Sommaire

Recap

Message aux élèves de l'ENSIIE

Si vous souhaitez compiler et tester votre code dans IntelliJ, vous devez noter que vous ne pouvez lancer votre code que si tout votre projet compile ! Déplacez donc les fichiers un par un.

Ensuite, le code d'éclipse utilise JUnit4 (et JUnit5) mais JUnit4 n'est pas nativement supporté dans IntelliJ donc il va falloir suivre les étapes dans la partie 4 (environ 3 minutes).

Vous devrez également suivre les étapes dans la partie 4, si après importation votre dossier src/ n'est pas bleu (ce qui est mauvais signe, mais est le comportement natif sous linux).

Intro

Introduction

IntelliJ IDEA est un IDE principalement pour le Java, avec un support pour JavaFX, R, OCaml et plein d'autres.

Leurs logiciels sont payants sauf IntelliJ Idea qui est gratuit. Étant étudiant, vous pouvez avoir tous leurs logiciels gratuitement.

Il y a un environ un logiciel par langage (environ 15 langages). Les interfaces sont toutes les mêmes, donc le changement est facile mais cela prend beaucoup de GO de stockage (~10GO pour 3 IDE).

Installation

- Allez sur https://www.jetbrains.com/shop/eform/students.
- Se créer un compte avec son adresse étudiante (@ensiie.fr)
- Activez votre compte...
- Téléchargez votre IDE (via https://www.jetbrains.com/toolbox-app/ par exemple)
- ou ici, si vous ne voulez pas le toolbox https://www.jetbrains.com/idea/download/
- Une fois lancé, connectez-vous pour activer votre licence (valide 1 an)

Avantages et inconvénients

✖ compiler une seule classe (compile tout ou rien)

✔ support JUnit5

✔ support Maven/Gradle

✔ support CodeWithMe

✔ autocompletion, correction orthographique, analyse du code

✔ base de données intégrée (version ultimate uniquement) et GIT

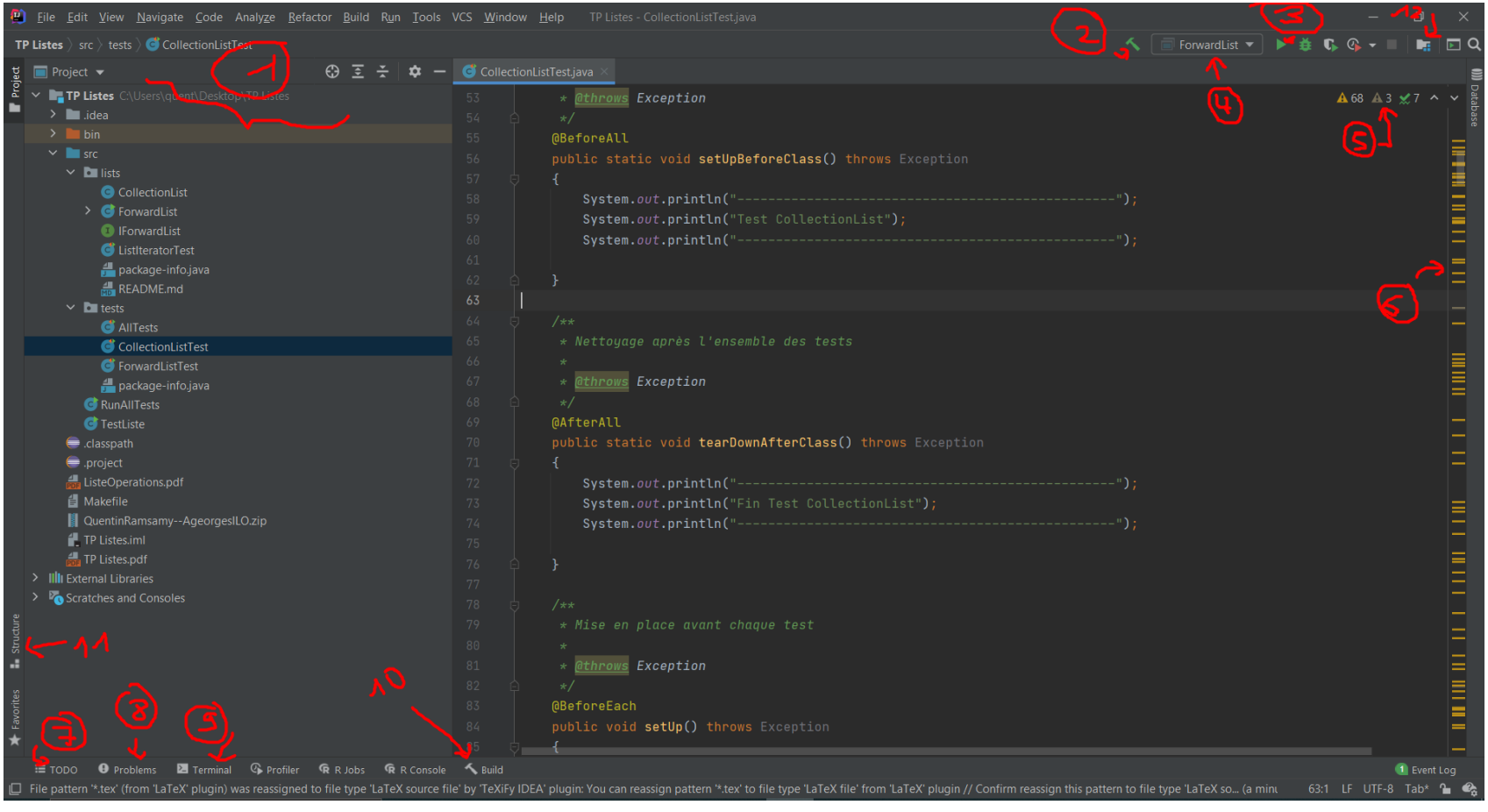
✔ Code Cleanup et génération de diagrammes UML

Lancement

- glisser-déposer un dossier contenant un projet pour qu'il l'importe
- ou créez un nouveau projet
  - sélectionnez **java**, puis un **JDK** (vous pouvez en télécharger un si vous n'en n'avez pas depuis IntelliJ)
  - un JDK vous permet de compiler et lancer JAVA (javac et java)
  - cliquez sur **next**, et cochez **"create project from template"** puis **next**
  - donner un **nom**, **dossier** et **un package** (pas important, vous pourrez le changer, **main** par exemple)
- avec PHPStorm, vous pouvez ouvrir un dossier se trouvant sur un serveur distant

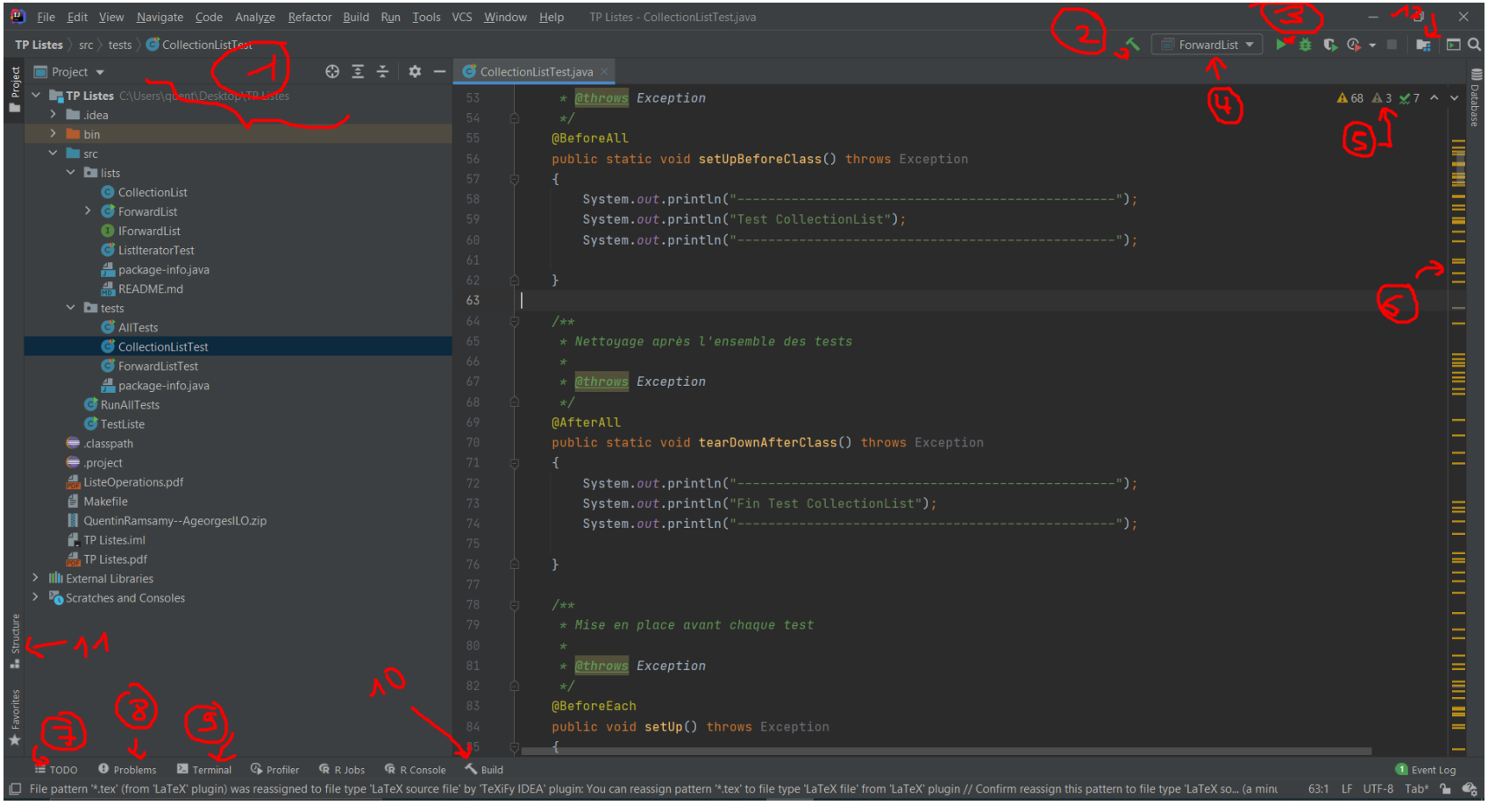
Interface

Interface



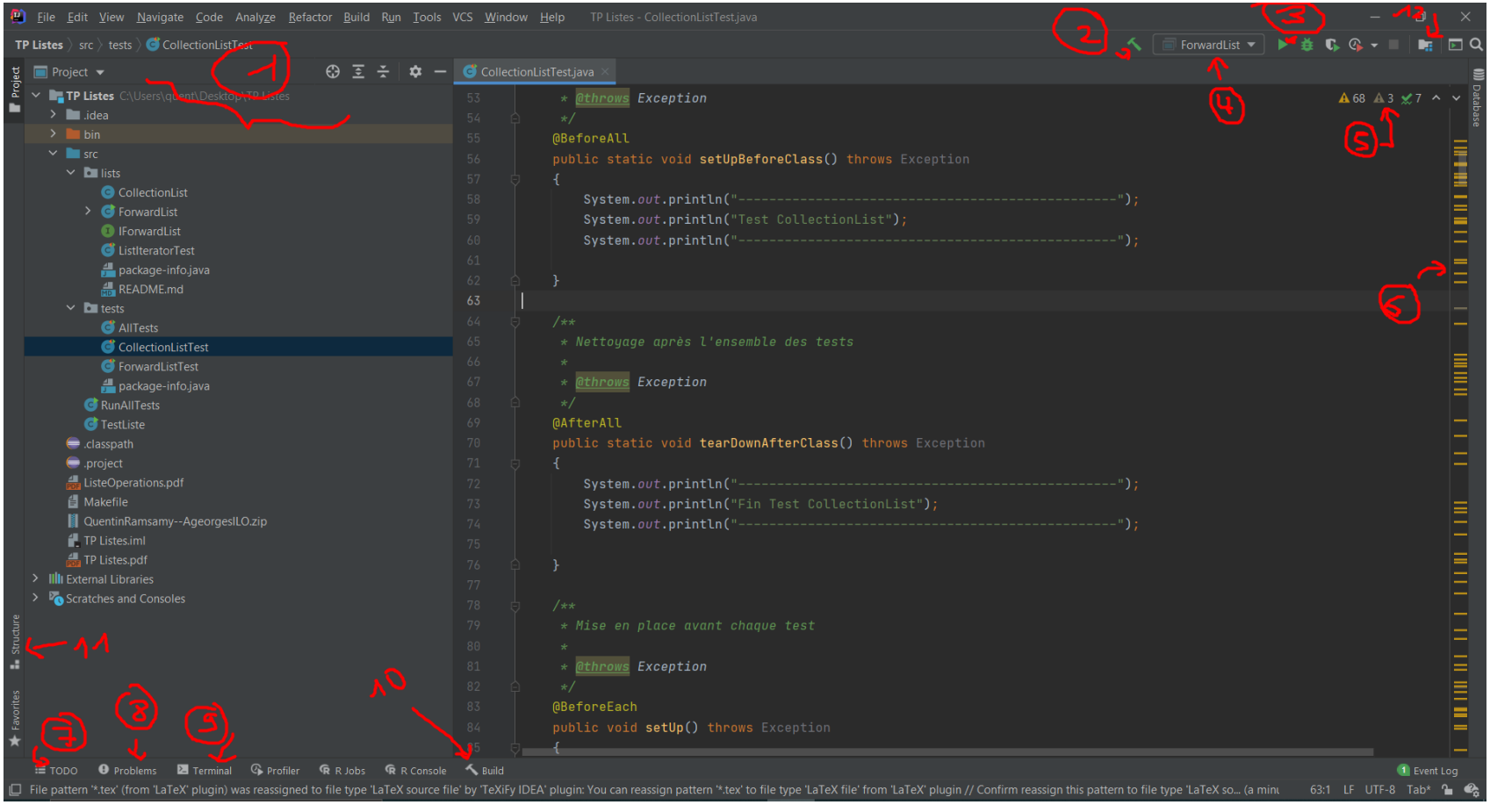
Voici l'interface générale. Vous avez globalement deux parties qui sont la **liste de vos fichiers** (1) et **vos fichiers ouverts** (ici seulement CollectionListTest.java).

Interface : compiler et lancer



- (2) compiler mais ne lance pas l'application
- (4) décider quoi compiler/lancer (quelle cible/classe/...)
- (3) lancer la classe choisie

Interface - erreurs/...



- (5) le nombre de warnings, ... dans le fichier actuel (cliquez dessus pour voir le détail)
- (6) vous avez un trait par ligne avec un warning, ...

Interface - erreurs/...

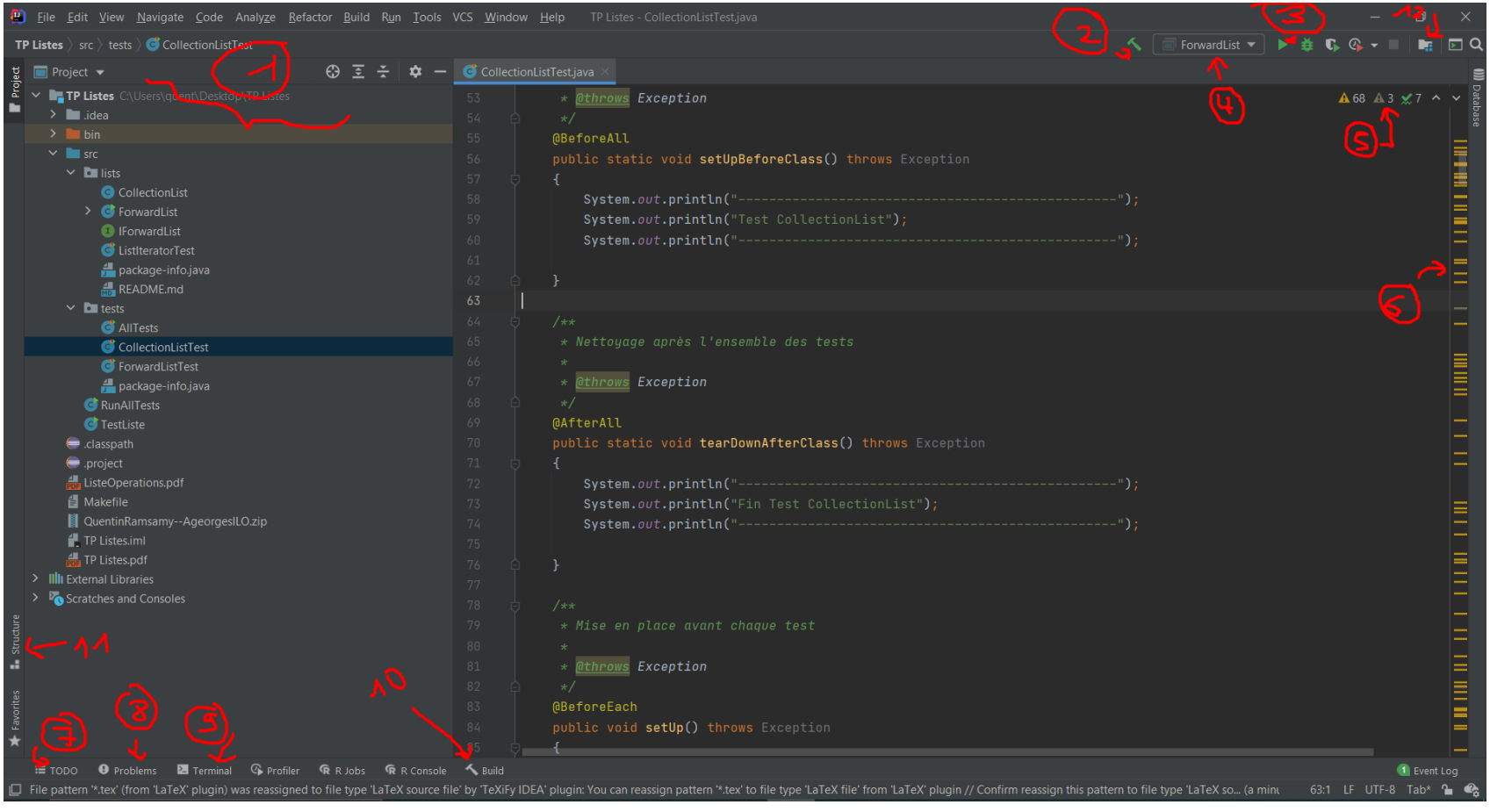
Les erreurs, les warnings, ... vous allez en avoir pleins ! C'est normal et le plus d'IDEA c'est qu'en plus de vous prévenir, il vous donne des corrections possibles à vos problèmes.

Il souligne un mot/méthode/classe/... en

- jaune : une possible erreur ou documentation manquante (warning)
- rouge : une erreur (danger)
- vert : une erreur typographique (typo)
- gris (pas souligné, mot en gris) : variable/méthode/classe non utilisée

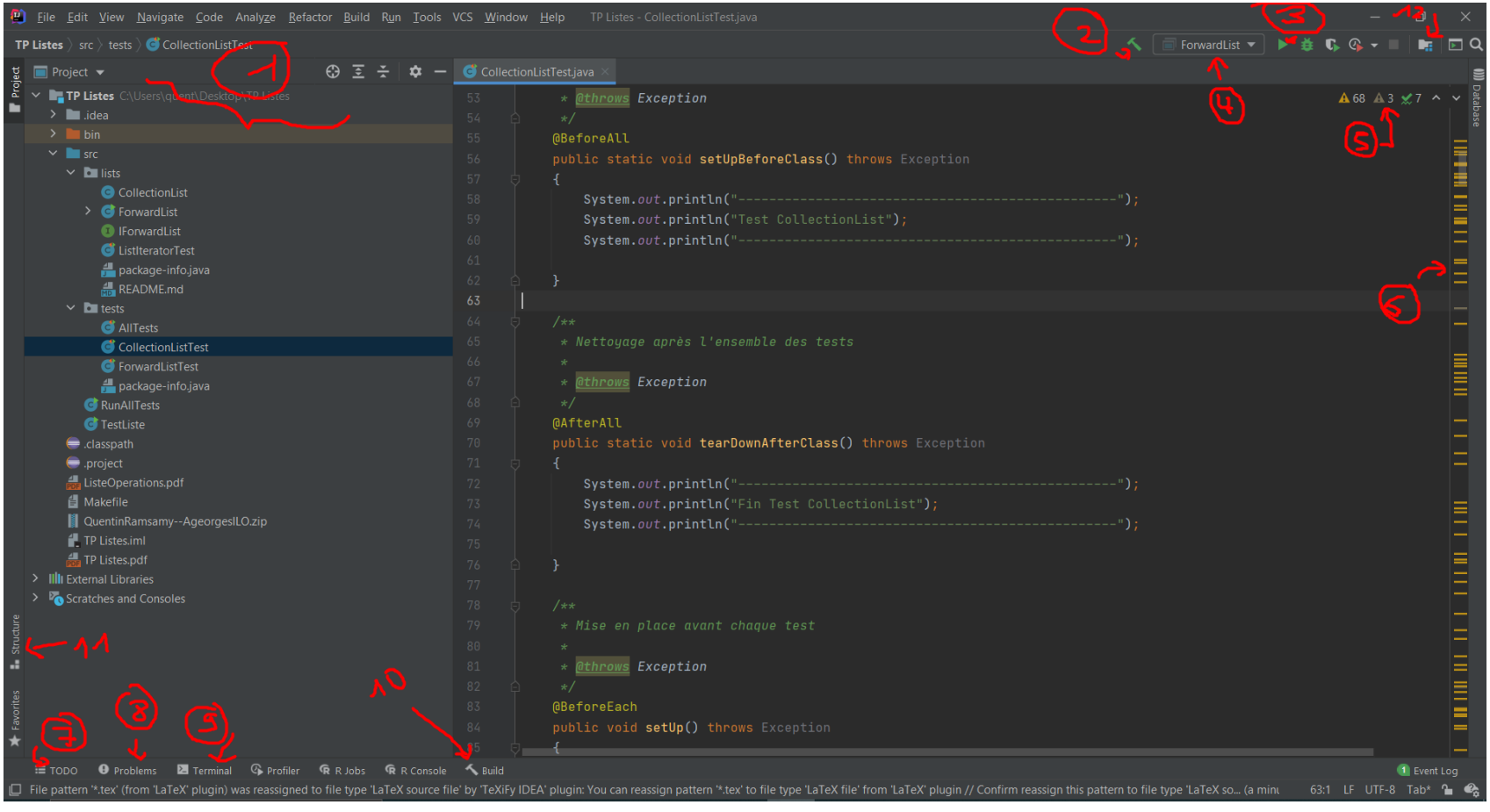
Vous pouvez faire **alt+entrée** dessus pour obtenir des suggestions pour résoudre le problème.

Interface : outils



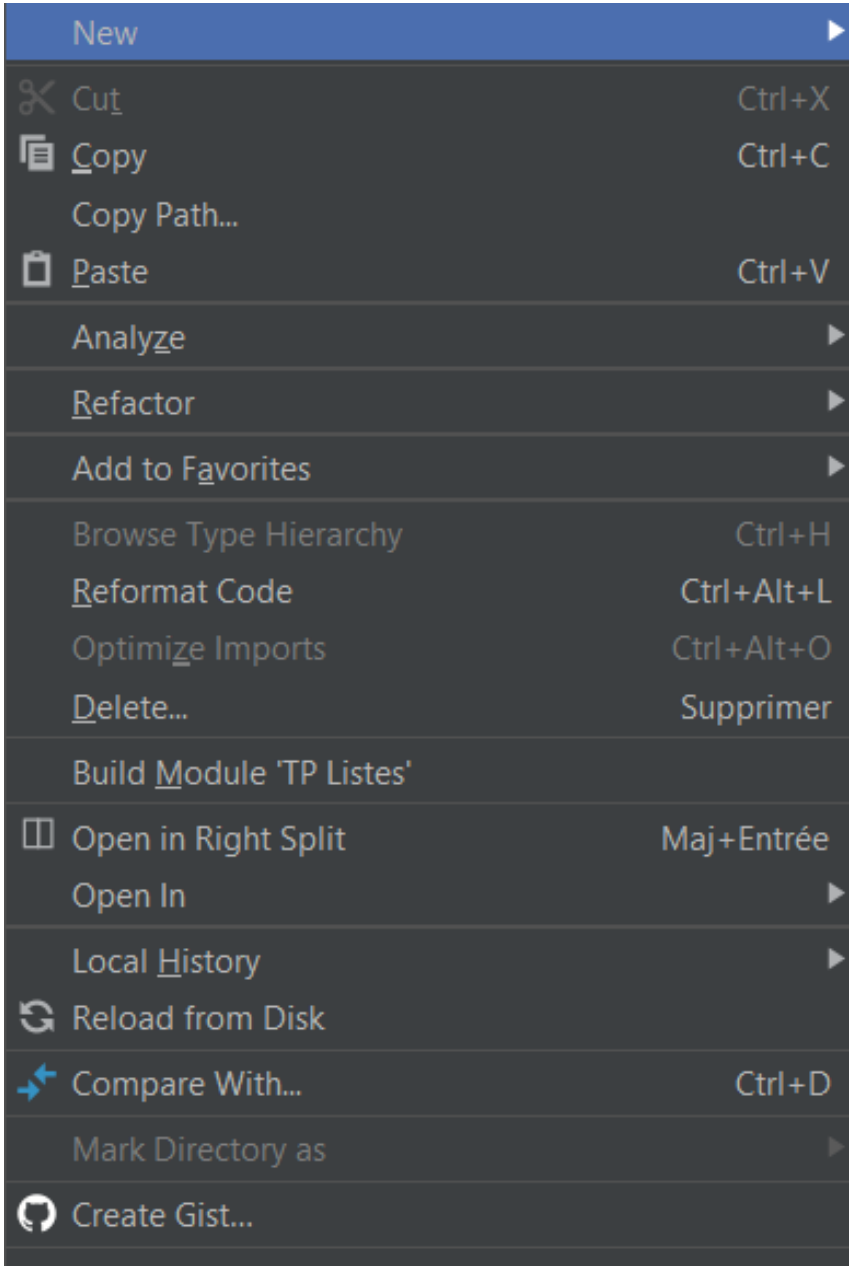
- (7) la liste des todo (ou fixme, commentaires spéciaux)
- (8) problèmes du fichier actuel
- (9) terminal
- (10) les messages à la compilation

Interface : le reste



- (11) la structure du fichier (en Java: les classes, les méthodes, ...)
- (12) project structure : gérer par exemple le JDK (java), les librairies...
- (?) non visible car pas de .git ici, mais vous avez un menu git

Interface : menu



- new : nouveau fichier
- analyse >
  inspect : voir les erreurs/...
- delete, copy, ...
- refactor
  : outil pour renommer
- reformat
  code : nettoyer votre code
- open in : ouvrir dans
  un explorateur de fichiers
  ou dans un terminal
- git : réinitialiser un
  fichier au dernier commit...

Raccourcis

Raccourcis basiques (1)

- **CTRL+F** : rechercher un mot dans le fichier actuel
- **CTRL+A** : tout sélectionner
- **CTRL+C** : copier
- **CTRL+V** : coller
- **CTRL+Z** : revenir en arrière
- **CTRL+SHIFT+Z** : annuler un retour en arrière

Raccourcis basiques (2)

- **SHIFT+[...]** : shift et une flèche/begin/end vous permet de rapidement sélectionner des trucs
- **CTRL+D** : si rien sélectionné duplique la ligne sinon duplique la sélection
- **CTRL+X** : si rien sélectionné supprime la ligne sinon supprime la sélection
- **CTRL+R** : faire du remplacement facilement
- **CTRL+G** : se déplacer à une ligne
- **CTRL+L** : se déplacer à la prochaine occurrence de la dernière recherche (avec CTRL-F ou CTRL-R)

Raccourcis pratiques

- SHIFT+SHIFT : outil de recherche (classe/méhode/fichier/paramètre)
- CTRL+E : fichiers récents et menus (git/...)
- CTRL+ALT+F : rechercher un mot dans tous les fichiers/un dossier/...
- CTRL+SHIFT+U : passer la sélection minuscule à majuscule et inversement
- ALT+7 : ouvre le menu de structure du fichier vu plus haut (11)
- CTRL+SHIFT+E : montre les 3 deniers endroits ou vous étiez

Orienté Objet (principalement)

- ALT+INSERT : générer les getters/setters/toString/equals/...
- CTRL+ALT+T : vous permet de générer un trycatch, un if, ... autour de la sélection
- CTRL+SHIFT+T : générer un test de la classe actuelle (donc faut être dans une classe)
- CTRL puis clic sur méthode/classe : vous montre les usages (appels, ...)

Project Structure

Project Structure : Utilité

Il s'agit de l'icône 12 de l'interface, pas loin de l'icône run et build.

Si jamais vous voulez **changer de JDK**, **importer des librairies** sans être dans un projet Maven/Gradle alors ce menu vous permettra de le faire.

En particulier, c'est dans ce menu que vous allez **définir le dossier des sources** (pour qu'IDEA puisse faire sont travail)... Vous aurez besoin de faire ça si le dossier contenant vos fichiers java n'est pas bleu.

Tutoriel complet si vous avez besoin des détails
https://memorize.lgs-games.com/fr/latest/utils/jetbrains/idea/ps.html

Project Structure : JUnit 3 et 4

En particulier, si vous importez un projet **Eclipse** utilisant JUnit 3 et 4 (pas besoin si vous utilisez uniquement 5), comme ceux de l'ENSIIE, vous devrez dans Project Structure

- allez dans librairies
- ajouter une librairie (+) puis maven (pas besoin de comprendre)
  - org.junit.platform:junit-platform-suite-api:1.7.0
  - org.junit.jupiter:junit-jupiter-engine:5.7.0
  - org.junit.platform:junit-platform-launcher:1.7.0
  - org.junit.vintage:junit-vintage-engine:5.7.0
  - org.junit.platform:junit-platform-runner:1.7.0
- allez dans Module > Dependancies et supprimer JUnit5

Pour être plus précis, il s'agit d'un patch que j'ai essayé de faire pour ne pas devoir utiliser des monstres (très pratiques) comme gradle et maven.

Tutoriel complet ici https://memorize.lgs-games.com/fr/latest/utils/jetbrains/idea/tests.html

Breakpoints et debug

Breakpoints et debug

Pour résumer, vous mettez des **breakpoint**, c'est à dire des pauses dans l'exécution du programme. Lorsque vous lancez le programme **en mode debug** (scarabée à côté de la flèche normale pour lancer), alors lorsqu'il rencontre un breakpoint, le programme se mets en pause et affiche les valeurs des variables...

Tutoriel complet ici https://memorize.lgs-games.com/fr/latest/utils/jetbrains/idea/debug.html

Pour aller plus loin

Pour aller plus loin

Ce cours n'est qu'un rapide aperçu d'informations pouvant être trouvées sur la page du cours
https://memorize.lgs-games.com/fr/latest/utils/jetbrains.html

En particulier, les points suivant n'ont pas étés abordés et le sont normalement sur le site

- gérer une base de données
- configuration (Snippets/...)
- utiliser le refactor
- générer l'UML
- utiliser GIT
- utiliser R, OCaml, ...