Jeu de société

Médéric HURIER

Master 1 STS - MIAGE

2012-2013

Sommaire

Présentation du jeu

Mon nom est Capitaine Duke

- Vous êtes le capitaine d'un vaisseau spatial
- Vous devez répondre aux requêtes votre équipage
- Pour chaque problème, il faut écrire une fonction Java qui vérifie des tests

Règles du jeu

- Chaque mission réussie vous récompense d'un nombre de points comptant pour votre score final
  - critères : difficulté de la partie, nombre d'essais restant, temps passé, dernier essai chanceux
- Si votre nombre d'essais descend en dessous de 0, la mission se termine et votre score est de 0
- A la fin des 6 scénarios, vous pouvez poster votre score sur un serveur dédié

Déroulement

- Lancer une nouvelle partie en choisissant la difficulté
- Liser l'histoire puis appuyer sur le bouton Commencer pour démarrer le chronomètre
- Écrire le code nécessaire et tester le
  - si la réponse est correcte, un message vous félicite et affiche votre score
  - sinon, une boîte de dialogue avec l'erreur s'affiche, avec la sortie du programme
- Les missions s'enchaînent ...

Démonstration

Choix d'implémentation

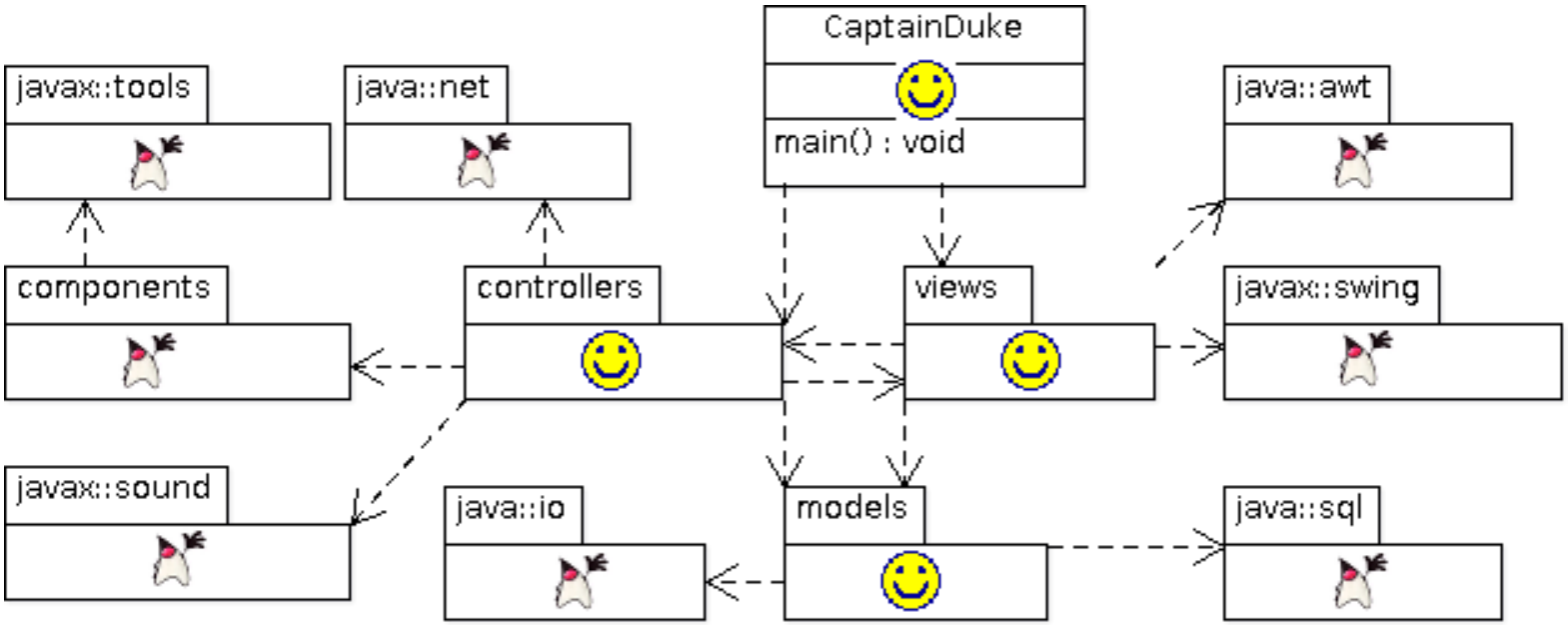
Les choix import.ants

- Interface graphique => ~~**JavaFX**~~ javax.swing
- Persistance => java.io.Serializable + java.sql
- Multimédia => javax.sound
- Méta-programmation => javax.tools.JavaCompiler + java.lang.reflect
- Réseau => java.net
- Multi-tâche => java.util.Timer

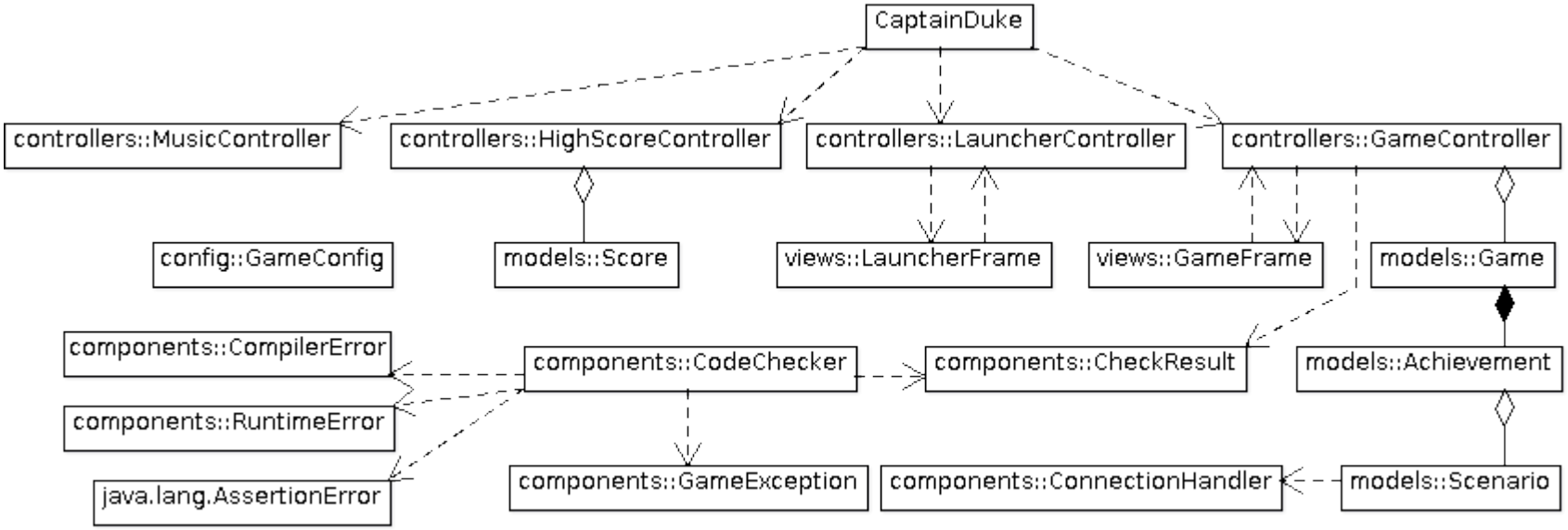
Beaucoup de bibliothèques natives.

exception : driver SQLite pour JDBC

Les paquets


CaptainDuke
main() : void
javax::tools
java::net
java::awt
components
controllers
views
javax::swing
javax::sound
java::io
models
java::sql

Les classes


CaptainDuke
controllers::MusicController
controllers::HighScoreController
controllers::LauncherController
controllers::GameController
config::GameConfig
models::Score
views::LauncherFrame
views::GameFrame
models::Game
components::CompilerError
components::CodeChecker
components::CheckResult
models::Achievement
components::RuntimeError
java.lang.AssertionError
components::GameException
components::ConnectionHandler
models::Scenario

En détail : la classe CodeChecker 1/4

```
private static File file(String code, String test) throws GameException {
  File javaFile;

  try {
    javaFile = File.createTempFile("javanaute", ".java");
    FileWriter out = new FileWriter(javaFile);
    String name = javaFile.getName().replace(".java", "");
    out.write("public class " + name + " {\n");
    out.write("public void test() {\n");
    out.write(test);
    out.write("\n}");
    out.write(code);
    out.write("\n}");
    out.close();
  } catch (IOException e) {
    throw new GameException("Impossible de créer le fichier temporaire");
  }

  return javaFile;

}
```

En détail : la classe CodeChecker 2/4

```
private static File compile(File file) throws CompilerError {
  ByteArrayOutputStream err = new ByteArrayOutputStream();
  JavaCompiler compiler = ToolProvider.getSystemJavaCompiler();
  File classFile = new File(file.getAbsolutePath().replace(".java", ".class"));
  int status = compiler.run(null, null, err, file.getAbsolutePath());
  String errStr = err.toString().replace(file.getAbsolutePath(), "");

  if (status != 0 || !classFile.exists()) {
    throw new CompilerError("COMPILATION\n" + errStr);
  }

  return classFile;

}
```

En détail : la classe CodeChecker 3/4

```
private static void run(File classFile) throws GameException, AssertionError, RuntimeError {
  String className = classFile.getName().replace(".class", "");
  String parentDir = classFile.getParent() + File.separator;
  URL[] searchPath = new URL[1];
  try {
    searchPath[0] = new File(parentDir).toURI().toURL();
  } catch (MalformedURLException ex) {
    throw new GameException("Impossible de résoudre le fichier class: " + ex);
  }

  try {
    ClassLoader cl = new URLClassLoader(searchPath);
    cl.setDefaultAssertionStatus(true);
    Class clazz = Class.forName(className, true, cl);
    Object object = clazz.newInstance();
    Method method = clazz.getDeclaredMethod("test");
    method.invoke(object);
  } catch (...) {
    throw new GameException("Impossible de charger la classe: " + ex);
  }

}
```

En détail : la classe CodeChecker 4/4

```
public static CheckResult check(String code, String test) throws GameException {
  if (test == null || test.isEmpty()) {
    return new CheckResult("", new CompilerError("La fonction de test est vide"));
  }

  File javaFile = null;
  File classFile = null;
  Error error = null;
  PrintStream oldOut = System.out;
  ByteArrayOutputStream newOut = new ByteArrayOutputStream();
  System.setOut(new PrintStream(newOut));
  try {
    javaFile = CodeChecker.file(code, test);
    classFile = CodeChecker.compile(javaFile);
    CodeChecker.run(classFile);
  } catch (...) {

     ...
  }

  System.setOut(oldOut);
  CheckResult result = new CheckResult(newOut.toString(), error);

  return result;

}
```

Conclusion

- Beaucoup de bibliothèques Java utilisées (SWING, multimédia, multi-tâche, HTTP, **méta-programmation**)
- Utilisation des concepts de Génie Logiciel (SCM, tests unitaires, refactorisation ...)
- Une grosse déception : JavaFX

Conclusion

- Beaucoup de bibliothèques Java utilisées (SWING, multimédia, multi-tâche, HTTP, **méta-programmation**)
- Utilisation des concepts de Génie Logiciel (SCM, tests unitaires, refactorisation ...)
- Une grosse déception : JavaFX

- Mais il m'en faudra plus pour aimer Java

Questions ?