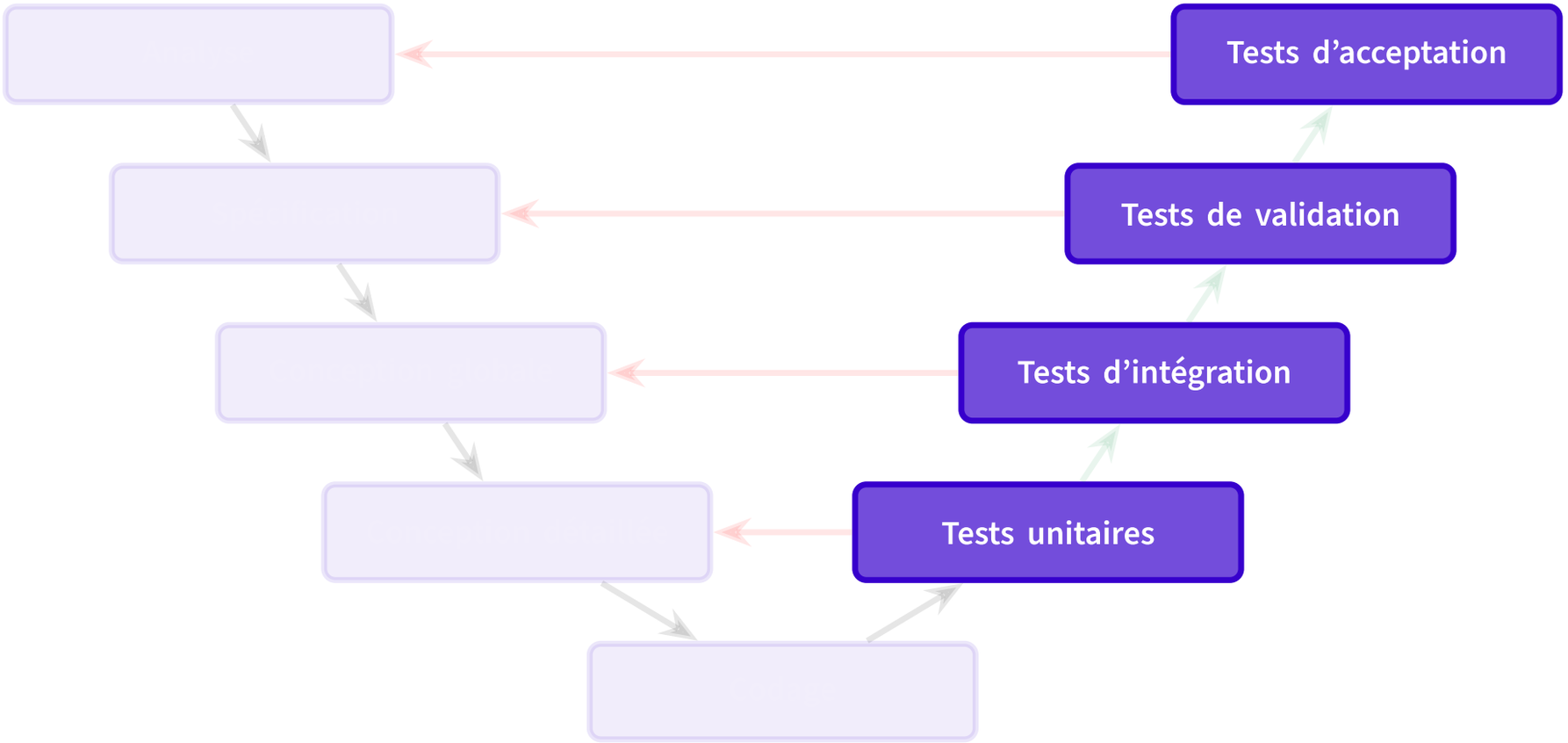
QUALITÉ DE DÉVELOPPEMENT

LES TESTS UNITAIRES EN JAVA

🎓 2A - Bachelor Universitaire de Technologie
🏛 IUT d'Orsay - Université Paris-Saclay - 2024/2025

Idir AIT SADOUNE
idir.ait-sadoune@universite-paris-saclay.fr

PROCESSUS DE DÉVELOPPEMENT EN V



LES ACTIVITÉS DE TESTS

Nous distinguons quatre types d'activités de tests :

1. **Les tests unitaires** : vérifier séparément le fonctionnement des modules d'un logiciel.
2. **Les tests d'intégration** : regrouper l'ensemble des modules et valider le fonctionnement global du logiciel.
3. **Les tests de validation** : valider la conformité du logiciel par rapport aux spécifications.
4. **Les tests d'acceptation** : vérifier avec le client que la solution proposée est conforme à ses exigences.

DANS CE COURS

Nous distinguons quatre types d'activités de tests :

1. **Les tests unitaires** : vérifier séparément le fonctionnement des modules d'un logiciel.
2. **Les tests d'intégration** : regrouper l'ensemble des modules et valider le fonctionnement global du logiciel.
3. **Les tests de validation** : valider la conformité du logiciel par rapport aux spécifications.
4. **Les tests d'acceptation** : vérifier avec le client que la solution proposée est conforme à ses exigences.

LA PHILOSOPHIE DU TEST

- Pas de différence entre test et débogage.
  - déboguer avec des jeux d'entrées
- Le test sert à montrer que le logiciel fonctionne.
  - exécuter une suite de tests sans défaillance
- Le test montre la présence d'erreurs mais pas leur absence.
  - permettre de diminuer le risque.

LES TESTS UNITAIRES

- **Un test unitaire** est **un script** permettant de **valider** le fonctionnement d'une partie d'un module.
  - ➠ une méthode, une class, une interface, ...
- En **Java**, le framework (JUnit), offre des fonctionnalités permettant le développement des tests unitaires.

LES ANNOTATIONS

JUnit utilise un système d'annotations définissant le comportement d'une méthode dans **une classe de test** :

- `@Test` : spécifie que la méthode annotée est un test.
- `@Before` ou `@BeforeEach` (JUnit5) : spécifie que la méthode annotée sera exécutée avant chaque méthode de test.
- `@BeforeClass` ou `@BeforeAll` (JUnit5): spécifie que la méthode annotée sera exécutée une fois avant toutes les méthodes de test.
- `@After` ou `@AfterEach` (JUnit5) : spécifie que la méthode annotée sera exécutée après chaque méthode de test.
- `@AfterClass` ou `@AfterAll` (JUnit5): spécifie que la méthode annotée sera exécutée une fois après toutes les méthodes de test.

LA CLASSE ASSERTIONS

Lien vers la documentation de la classe Assertions

LA CLASSE ASSERTIONS

- `void assertTrue(boolean condition)`
  vérifie qu'une condition est vraie.
- `void assertFalse(boolean condition)`
  vérifie qu'une condition est fausse.
- `void assertNotNull(Object object)`
  vérifie qu'un objet n'est pas null.
- `void assertNull(Object object)`
  vérifie qu'un objet est null.
- `void assertEquals(Object expected, Object actual)`
  vérifie l'égalité entre deux objets ou deux types primitifs.

LA CLASSE ASSERTIONS

- ...
- `void assertSame(Object expected, Object actual)`
  vérifie l'égalité entre deux références d'objets.
- `void assertNotSame(Object expected, Object actual)`
  vérifie que deux références d'objets sont différentes.
- `void assertArrayEquals(expectedArray, resultArray)`
  vérifie que deux tableaux sont égaux.
- `void fail(String)`
  force un test à l'échec.

CODE JAVA À TESTER

```java
public class Contact {
    private String firstName;
    private String lastName;
    private String phoneNumber;

    public Contact(String firstName, String lastName, String phoneNumber) {
        this.firstName = firstName;
        this.lastName = lastName;
        this.phoneNumber = phoneNumber;
    }

    public String getFirstName() {
        return firstName;
    }

    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }

    public String getLastName() {
        return lastName;
    }

    public void setLastName(String lastName) {
```

CODE JAVA À TESTER

```java
import java.util.Collection;
import java.util.Map;
import java.util.concurrent.ConcurrentHashMap;

public class ContactManager {

    Map<String, Contact> contactList = new ConcurrentHashMap<String, Contact>();

    public void addContact(String firstName, String lastName, String phoneNumber) {
        Contact contact = new Contact(firstName, lastName, phoneNumber);
        this.validateContact(contact);
        this.checkIfContactAlreadyExist(contact);
        this.contactList.put(this.generateKey(contact), contact);
    }

    public Collection<Contact> getAllContacts() {
        return this.contactList.values();
    }

    private void checkIfContactAlreadyExist(Contact contact) {
        if (this.contactList.containsKey(this.generateKey(contact)))
            throw new RuntimeException("Contact Already Exists");
    }

```

PREMIER TEST JUNIT5

```java
import org.junit.jupiter.api.*;
import static org.junit.jupiter.api.Assertions.*;

public class ContactManagerTest {

  @Test
  public void shouldCreateContact() {
      ContactManager contactManager = new ContactManager();
      contactManager.addContact("John", "Doe", "0123456789");

      assertTrue(! contactManager.getAllContacts().isEmpty());
      assertFalse(contactManager.getAllContacts().isEmpty());
      assertEquals(1, contactManager.getAllContacts().size());
  }
}
```

CYCLE DE VIE D'UN TEST JUNIT

```java
import org.junit.jupiter.api.*;
import org.junit.jupiter.api.condition.EnabledOnOs;
import org.junit.jupiter.api.condition.OS;
import org.junit.jupiter.params.ParameterizedTest;
import org.junit.jupiter.params.provider.CsvFileSource;
import org.junit.jupiter.params.provider.CsvSource;
import org.junit.jupiter.params.provider.MethodSource;
import org.junit.jupiter.params.provider.ValueSource;

import java.util.Arrays;
import java.util.List;

import static org.junit.jupiter.api.Assertions.assertEquals;
import static org.junit.jupiter.api.Assertions.assertFalse;

public class ContactManagerTest {

    private ContactManager contactManager;

    @BeforeAll
    public static void setupAll() {
        System.out.println("Should Print Before All Tests");
    }

```

RÉPÉTER DES TESTS AVEC JUNIT

```java
import org.junit.jupiter.api.*;
import org.junit.jupiter.api.condition.EnabledOnOs;
import org.junit.jupiter.api.condition.OS;
import org.junit.jupiter.params.ParameterizedTest;
import org.junit.jupiter.params.provider.CsvFileSource;
import org.junit.jupiter.params.provider.CsvSource;
import org.junit.jupiter.params.provider.MethodSource;
import org.junit.jupiter.params.provider.ValueSource;

import java.util.Arrays;
import java.util.List;

import static org.junit.jupiter.api.Assertions.assertEquals;
import static org.junit.jupiter.api.Assertions.assertFalse;

public class ContactManagerTest {

    private ContactManager contactManager;

    @BeforeAll
    public static void setupAll() {
        System.out.println("Should Print Before All Tests");
    }

```

LES TESTS PARAMÉTRÉS

```java
import org.junit.jupiter.api.*;
import org.junit.jupiter.api.condition.EnabledOnOs;
import org.junit.jupiter.api.condition.OS;
import org.junit.jupiter.params.ParameterizedTest;
import org.junit.jupiter.params.provider.CsvFileSource;
import org.junit.jupiter.params.provider.CsvSource;
import org.junit.jupiter.params.provider.MethodSource;
import org.junit.jupiter.params.provider.ValueSource;

import java.util.Arrays;
import java.util.List;

import static org.junit.jupiter.api.Assertions.assertEquals;
import static org.junit.jupiter.api.Assertions.assertFalse;

public class ContactManagerTest {

    private ContactManager contactManager;

    @BeforeAll
    public static void setupAll() {
        System.out.println("Should Print Before All Tests");
    }

```

LES TESTS IGNORÉS

```java
import org.junit.jupiter.api.*;
import org.junit.jupiter.api.condition.EnabledOnOs;
import org.junit.jupiter.api.condition.OS;
import org.junit.jupiter.params.ParameterizedTest;
import org.junit.jupiter.params.provider.CsvFileSource;
import org.junit.jupiter.params.provider.CsvSource;
import org.junit.jupiter.params.provider.MethodSource;
import org.junit.jupiter.params.provider.ValueSource;

import java.util.Arrays;
import java.util.List;

import static org.junit.jupiter.api.Assertions.assertEquals;
import static org.junit.jupiter.api.Assertions.assertFalse;

public class ContactManagerTest {

    private ContactManager contactManager;

    @BeforeAll
    public static void setupAll() {
        System.out.println("Should Print Before All Tests");
    }

```

TESTS DES EXCEPTIONS AVEC JUNIT

Utilisation de la méthode `assertThrows`

```java
import org.junit.jupiter.api.*;
import static org.junit.jupiter.api.Assertions.*;

public class ContactManagerTest {

@Test
public void shouldThrowRuntimeExceptionWhenFirstNameIsNull() {
    ContactManager contactManager = new ContactManager();

    Assertions.assertThrows(RuntimeException.class, () -> {
        contactManager.addContact(null, "Doe", "0123456789");
    });
}
}
```

MERCI

Version PDF des slides

Retour à l'accueil - Retour au plan