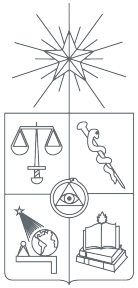


dcc
CIENCIAS DE LA COMPUTACIÓN
UNIVERSIDAD DE CHILE

Auxiliar 3:
Git, Visibilidad y Testing

Tomas Vallejos
tomas.vallejos@ing.uchile.cl

Agenda

- Repaso
- P *** R
- Testing
- Visibilidad
- Ejercicios

Repaso: this y super

- ¿ Qué son this y super ?

Repaso: this y super

- ¿ Qué son this y super ?
- ¿ Qué es una pseudo-variable ?

Repaso: Clase abstracta vs Interfaz

- ¿ Diferencias ?

Repaso: Clase abstracta vs Interfaz

- ¿ Diferencias ?
- ¿ Cuando se usa cada una ?

Repaso: Clase abstracta vs Interfaz

- ¿ Diferencias ?
- ¿ Cuando se usa cada una ?
- ¿ Se usan como tipo ?

Repaso: Clase abstracta vs Interfaz

- ¿ Diferencias ?
- ¿ Cuando se usa cada una ?
- ¿ Se usan como tipo ?
- ¿ Se usan para reutilizar código ?

P *** R (1/2)

Git Basics

1. `$ git add a`
2. `$ git commit -m “commit a”`
3. `$ git push`

P *** R (2/2)

Pull Request

1. `$ git checkout -b nuevaRama`
2. Hacer cambios
3. `$ git push --set-upstream origin nuevaRama`
4. Ir a branches, new pull request
5. Merge

Testing: JUnit 5

- @BeforeAll
- @BeforeEach
- @Test
- @RepeatedTest
- @AfterEach
- @AfterAll

Testing: JUnit 5

- **@BeforeAll: Antes de todos los test**
- **@BeforeEach: Antes de cada test**
- @Test
- @RepeatedTest
- @AfterEach
- @AfterAll

Testing: JUnit 5

- @BeforeAll: Antes de todos los test
- @BeforeEach: Antes de cada test
- **@Test: Declara que el método es un test**
- **@RepeatedTest: Declara que el test se va a repetir**
- @AfterEach
- @AfterAll

Testing: JUnit 5

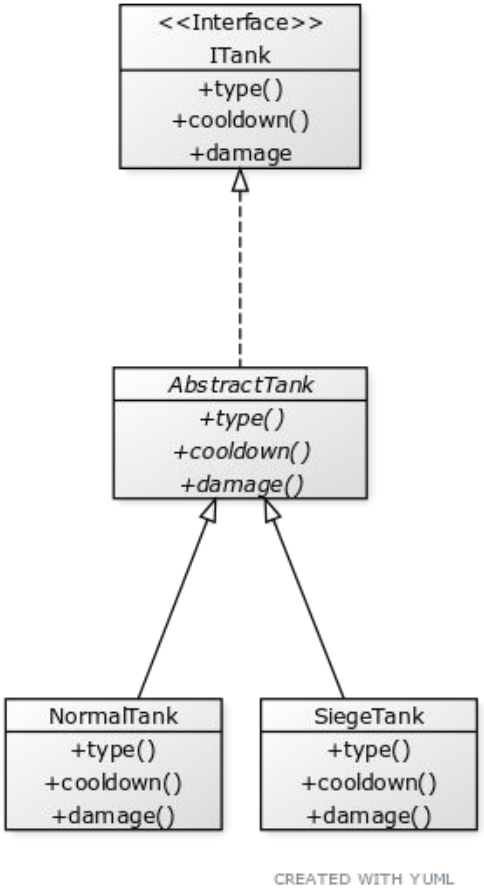
- @BeforeAll: Antes de todos los test
- @BeforeEach: Antes de cada test
- @Test: Declara que el método es un test
- @RepeatedTest: Declara que el test se va a repetir
- **@AfterEach: Después de cada test**
- **@AfterAll: Después de todos los test**

Testing

	Cooldown	Damage
Normal	0.74	15
Siege	2.14	40

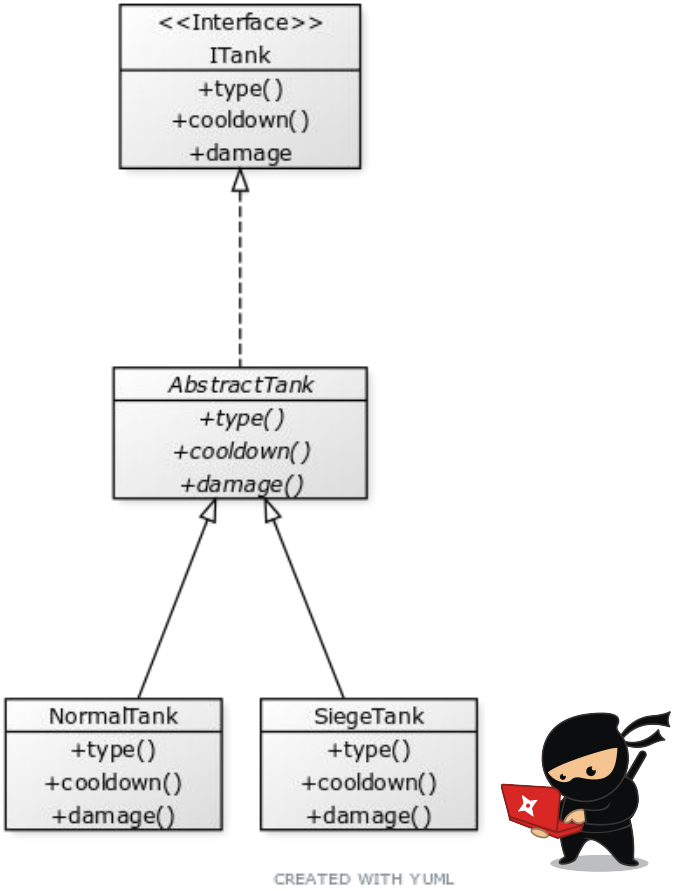
Testing

	Cooldown	Damage
Normal	0.74	15
Siege	2.14	40



Testing

	Cooldown	Damage
Normal	0.74	15
Siege	2.14	40



Testing

Las clases de test son **clases:**

- Pueden implementar interfaces
- Pueden ser abstractas
- Pueden extender clases

Testing

Las clases de test son **clases:**

- Pueden implementar interfaces
- Pueden ser abstractas
- Pueden extender clases

V • T • E	**History of the tank** [hide]
Era	World War I • Interwar • World War II • Cold War • Post-Cold War
Country	Australia • United Kingdom • Cuba • China • Canada • New Zealand • Czechoslovakia • France • Germany • Iran • Iraq • Italy • Israel • Japan • Poland • North Korea • South Korea • Soviet Union • Spain • United States
Type	Light tank • Medium tank • Heavy tank • Super-heavy tank • Cruiser tank • Infantry tank • Main battle tank • Tank destroyer • Tankette • Assault gun • Self-propelled gun • Self-propelled anti-aircraft weapon • Self-propelled artillery • Self-propelled mortar • Multiple rocket launcher
	Tanks portal

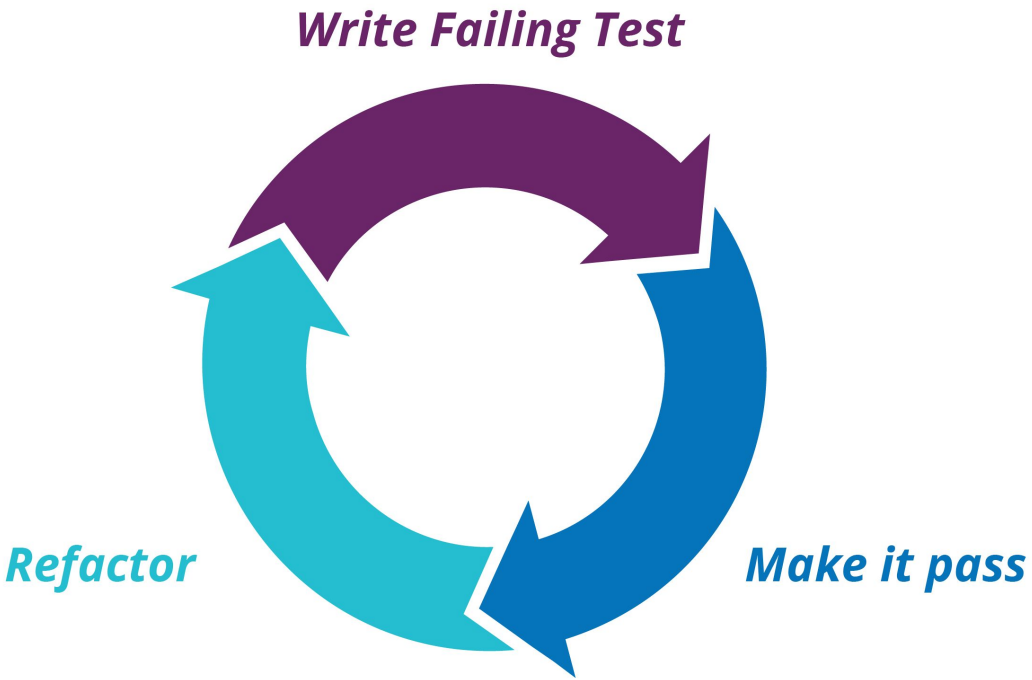
Testing

Las clases de test son **clases:**

- Pueden implementar interfaces
- Pueden ser abstractas
- Pueden extender clases

V • T • E	**History of the tank** [hide]
Era	World War I • Interwar • World War II • Cold War • Post-Cold War
Country	Australia • United Kingdom • Cuba • China • Canada • New Zealand • Czechoslovakia • France • Germany • Iran • Iraq • Italy • Israel • Japan • Poland • North Korea • South Korea • Soviet Union • Spain • United States
Type	Light tank • Medium tank • Heavy tank • Super-heavy tank • Cruiser tank • Infantry tank • Main battle tank • Tank destroyer • Tankette • Assault gun • Self-propelled gun • Self-propelled anti-aircraft weapon • Self-propelled artillery • Self-propelled mortar • Multiple rocket launcher
	Tanks portal

Test Driven Development (TDD)

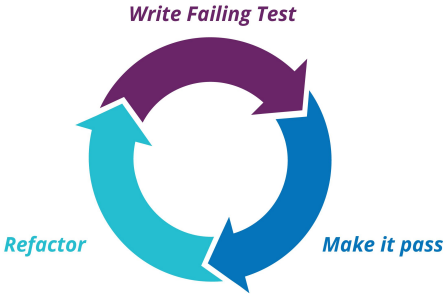


Test Driven Development (TDD)

Una cosa a la vez.

1. No se puede producir código sin tener un test que falle
2. No se puede escribir más que lo necesario para que el test falle, no compilar es fallar
3. No se puede escribir más código del necesario para que el test pase

Test Driven Development (TDD)



Una cosa a la vez.

1. No se puede producir código sin tener un test que falle
2. No se puede escribir más que lo necesario para que el test falle, no compilar es fallar
3. No se puede escribir más código del necesario para que el test pase

Visibilidad 1

```
public class A {
    private String method1() {return "A.method1()";}

    public String method2() {return "A.method2() > " + this.method1();}
}

public class B extends A {
    public String method1() {return "B.method1()";}

    public static void main(String[] args) {
        System.out.println(new B().method2());
    }
}
```

Visibilidad 2

```
public class C {
    protected String method1() {return "C.method1()";}

    public String method2() {return "C.method2() > " + this.method1();}
}

public class D extends C {
    public String method1() {return "D.method1()";}

    public static void main(String[] args) {
        System.out.println(new D().method2());
    }
}
```

WARNING
BOSS BATTLE

Visibilidad 3

```
public class A {
	private String method1() {
	return "A.method1()";}

	public String method2() {
	return "A.method2() > " + this.method1();}
}
public class B extends A {
	public String method1() {
		return "B.method1()";
	}
}
public class C {
	protected String method1() {
		return "C.method1()";
	}
	public String method2() {
		return "C.method2() > " + this.method1();
	}
}
public class D extends C {
	public String method1() {
		return "D.method1()";
	}
}
```

```
public class E {
	public String method1() {
		return "E.method1()";
	}
	public String method2() {
		return "E.method2() > "+this.method1();
	}
}
public class F extends E {
	public String method1() {
		return "F.method1()";
	}
}

public class G {

	public static void main(String[] args) {
		System.out.println(new A().method2());
		System.out.println(new B().method2());
		System.out.println(new C().method2());
		System.out.println(new D().method2());
		System.out.println(new E().method2());
		System.out.println(new F().method2());
	}
}
```

Visibilidad 3

```
public class A {
	private String method1() {
	return "A.method1()";}

	public String method2() {
	return "A.method2() > " + this.method1();}
}
public class B extends A {
	public String method1() {
		return "B.method1()";
	}
}
public class C {
	protected String method1() {
		return "C.method1()";
	}
	public String method2() {
		return "C.method2() > " + this.method1();
	}
}
public class D extends C {
	public String method1() {
		return "D.method1()";
	}
}
```

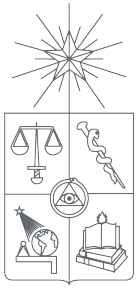
```
public class E {
	public String method1() {
		return "E.method1()";
	}
	public String method2() {
		return "E.method2() > "+this.method1();
	}
}
public class F extends E {
	public String method1() {
		return "F.method1()";
	}
}

public class G {

	public static void main(String[] args) {
		System.out.println(new A().method2());
		System.out.println(new B().method2());
		System.out.println(new C().method2());
		System.out.println(new D().method2());
		System.out.println(new E().method2());
		System.out.println(new F().method2());
	}
}
```

Accessibility

```
public class Animal {
     private String name;
     public Animal(String name) {
          this.name = name;
     }

     private String getName() {
          return name;
     }

     public String getPair(Animal paired) {
          return this.getName() + " with " + paired.getName();
     }

     public static void main(String[] args) {
          System.out.println(new Animal("Jirafa").getPair(new Animal("Antilope")));
          System.out.println(new Animal("Tigre").getName());
     }
}
```

Ejercicio Herencia, Interfaz y CA (propuesto)

https://classroom.github.com/a/4cCdLZkh



dcc
CIENCIAS DE LA COMPUTACIÓN
UNIVERSIDAD DE CHILE

Auxiliar 3:
Git, Visibilidad y Testing

Tomas Vallejos
tomas.vallejos@ing.uchile.cl