Call of Rum

Projet ILU4

L’organisation

NDOYE Assane - Chef de Projet et Responsable Tests Fonctionnels

RICHARD Jérémy - Responsable Développement Noyau

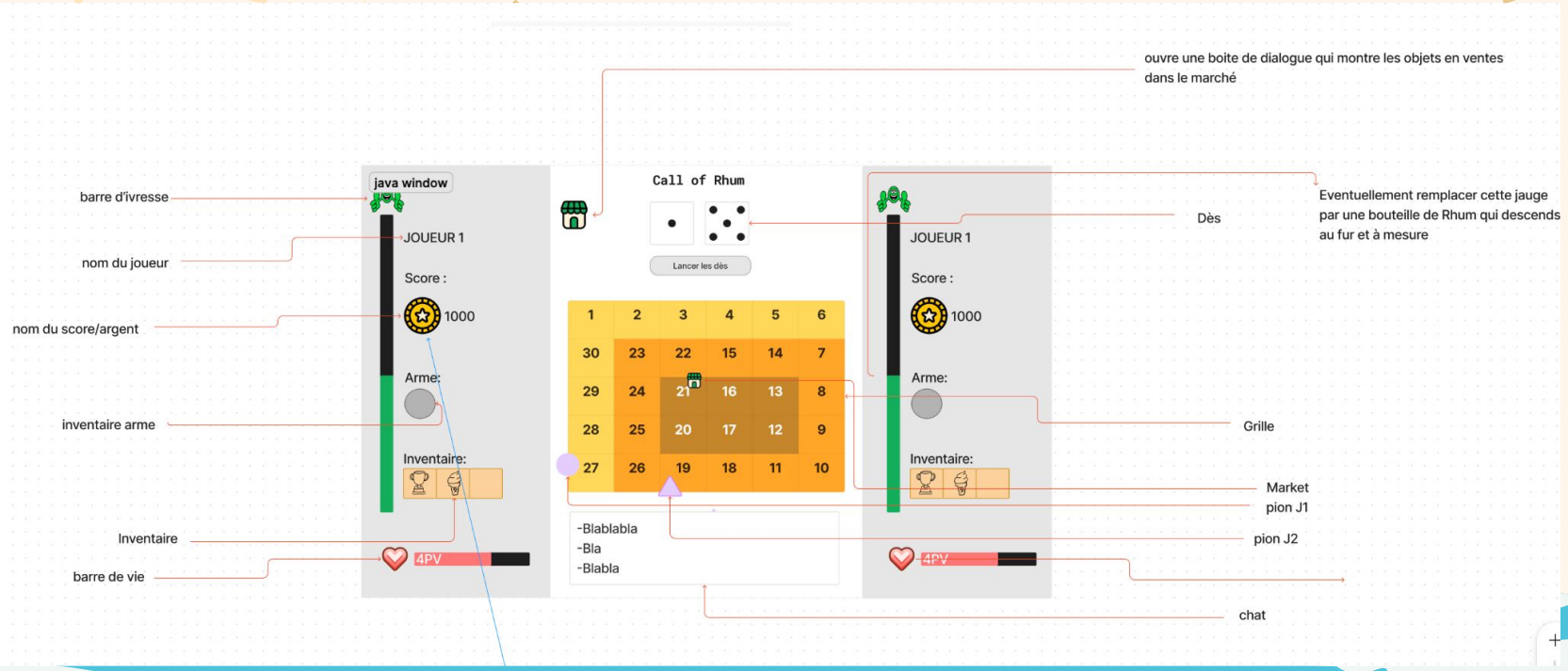
JOSEPH Wilkens Marc Johnley - Responsable Spécifications

IVANOVA Alina - Responsable Modélisation

FRANCES-GENTILLET Solène - Responsable IHM

LAFORGE Matéo - Responsable Technique

Présentation du jeu



Présentation du jeu

Combat

Comment lancer un combat ?

Duel si les deux pirates sont sur la même case

Qui gagne le combat ?

Pile ou face pondéré lors des combats

Comment améliorer ses chances de gagner un duel?

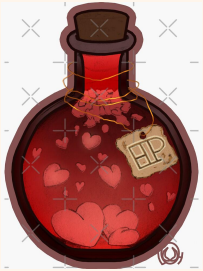
En récupérant des armes dans le shop

Case spéciale (Marché)

restaure les pv mais diminue les chances de gagner un combat)

armes (chance de gagner un combat / dégâts / vol)

Case spéciale (Coffre)



Potion de vie redonne de la vie

pierre de lucidité (ignore les effets de l’alcool)

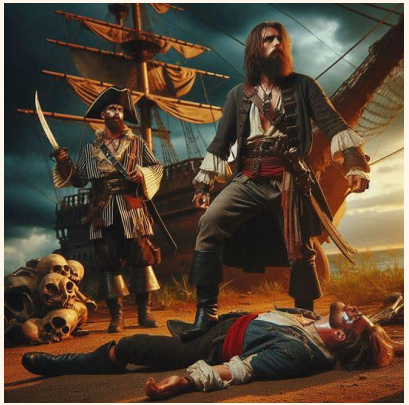
trèfle augmente les chance de l’or 150% d’or

poudre à canon (150% de chance de gagner un combat)

Case speciale (bonus/malus)

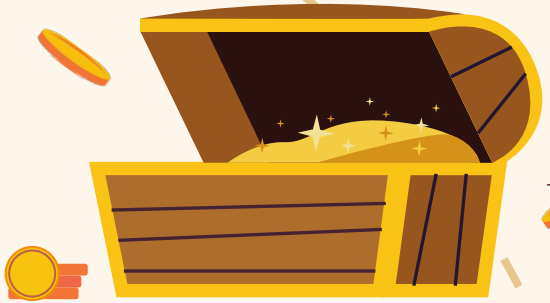
Qui gagne ?

Trouver le grand trésor caché



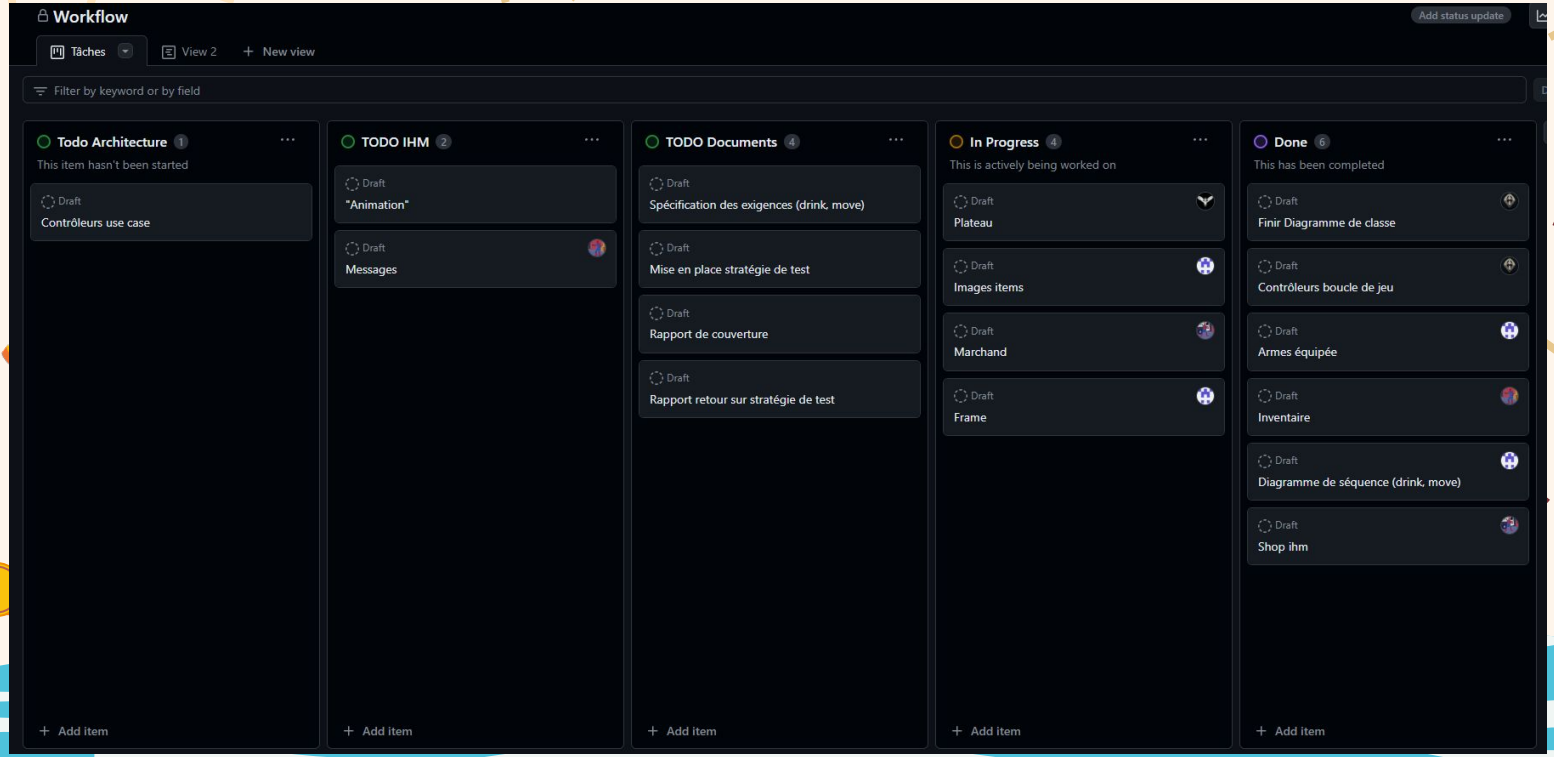
Tuer son adversaire

Organisation git

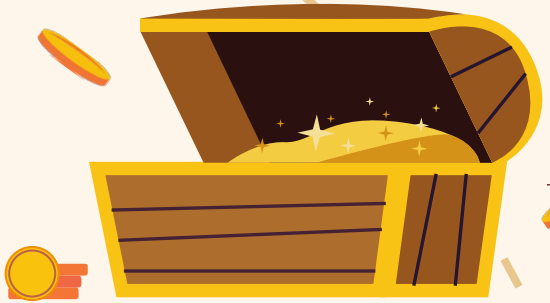


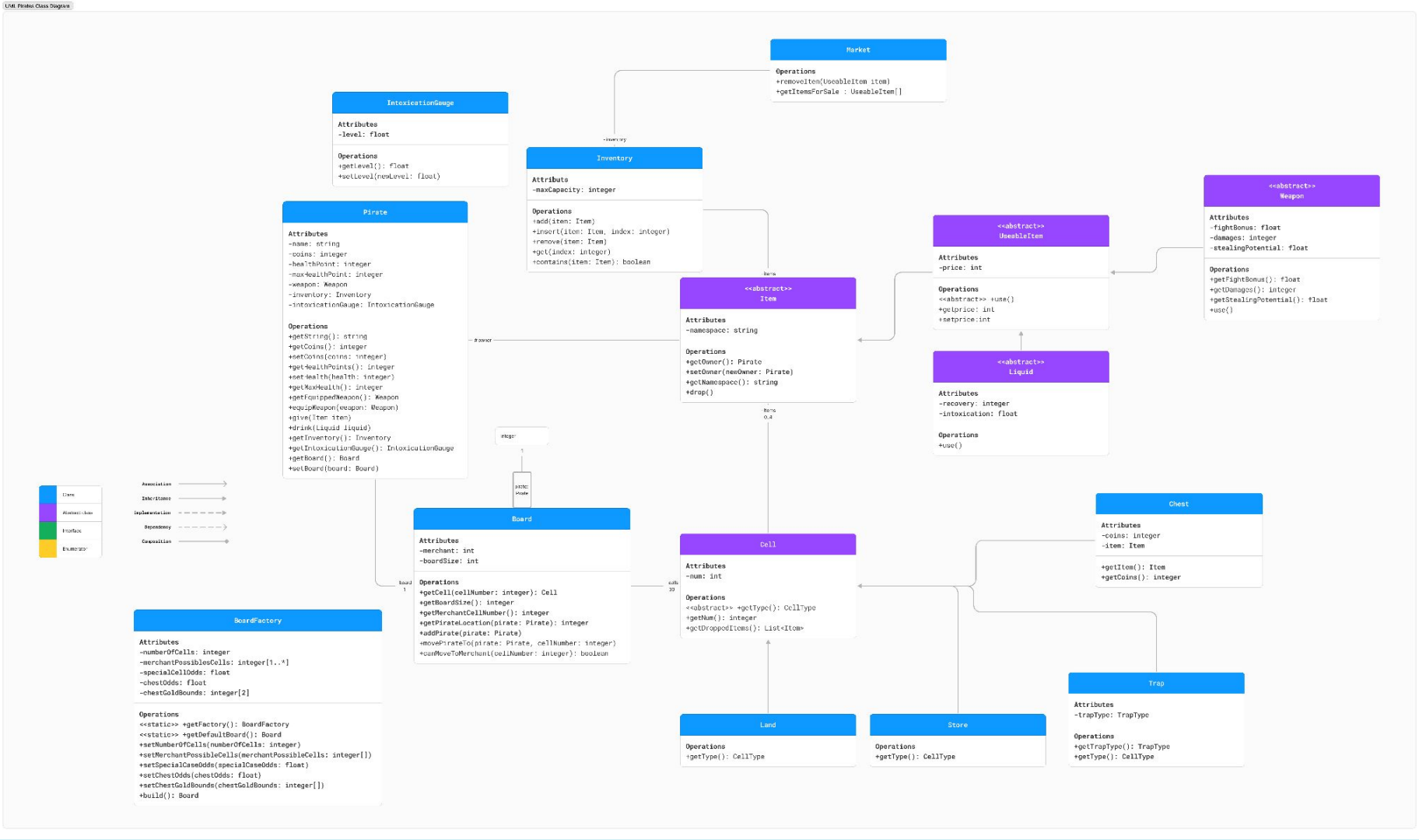
Organisation git

Organisation git

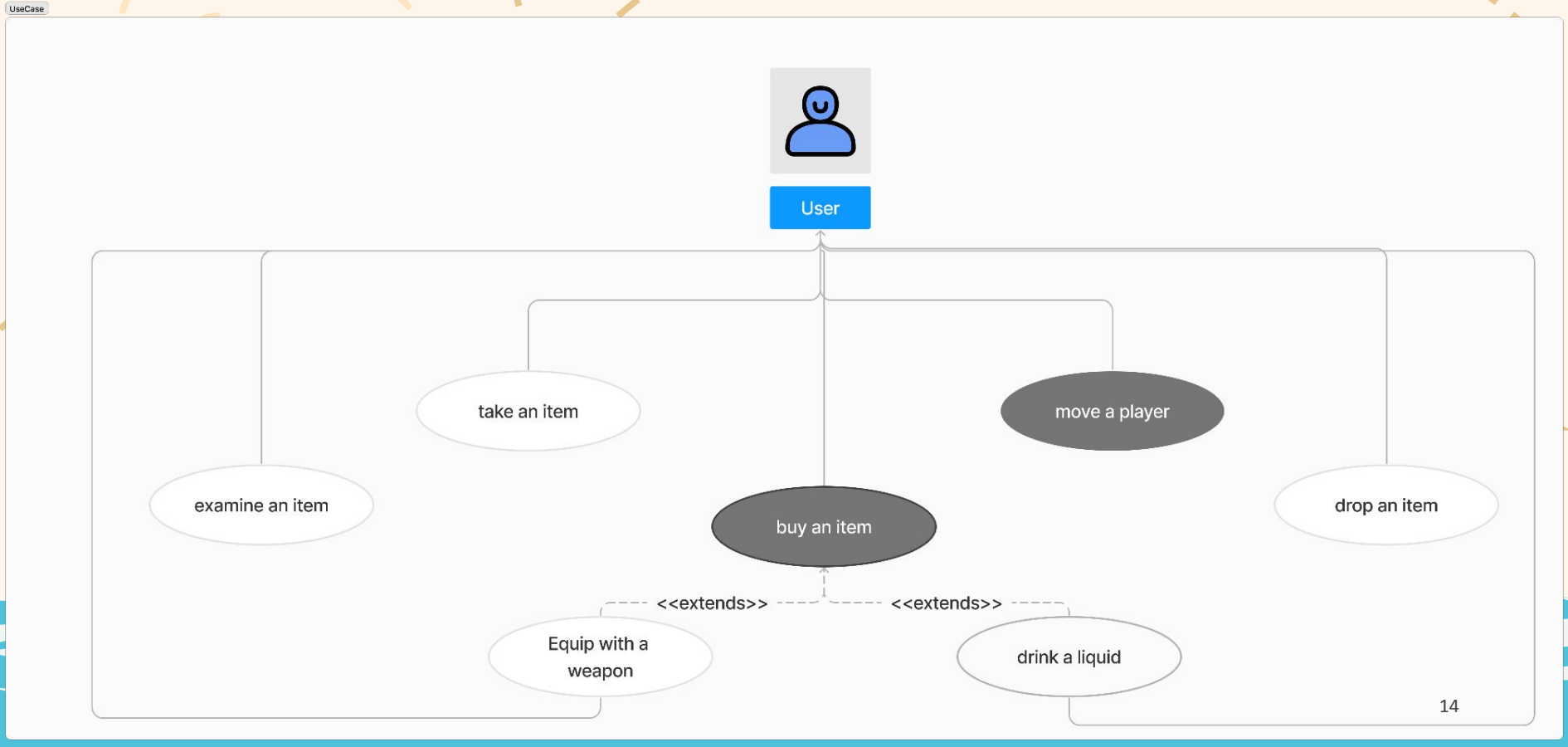


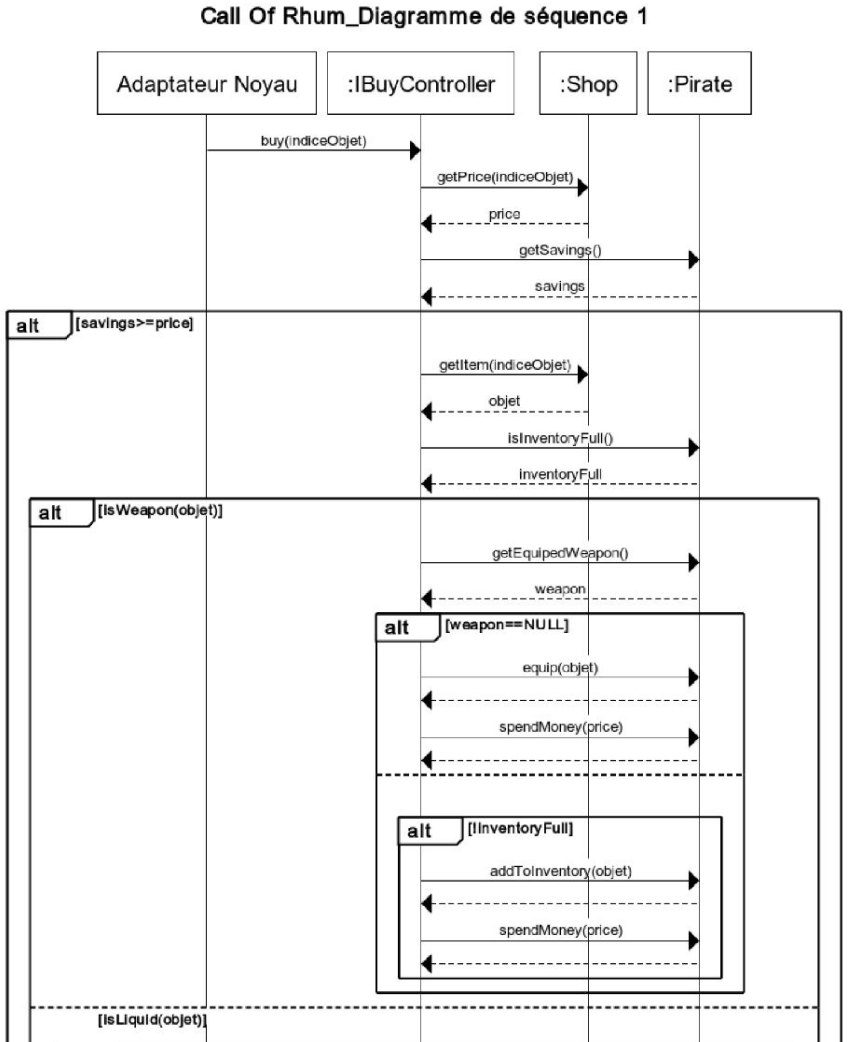
Modèle

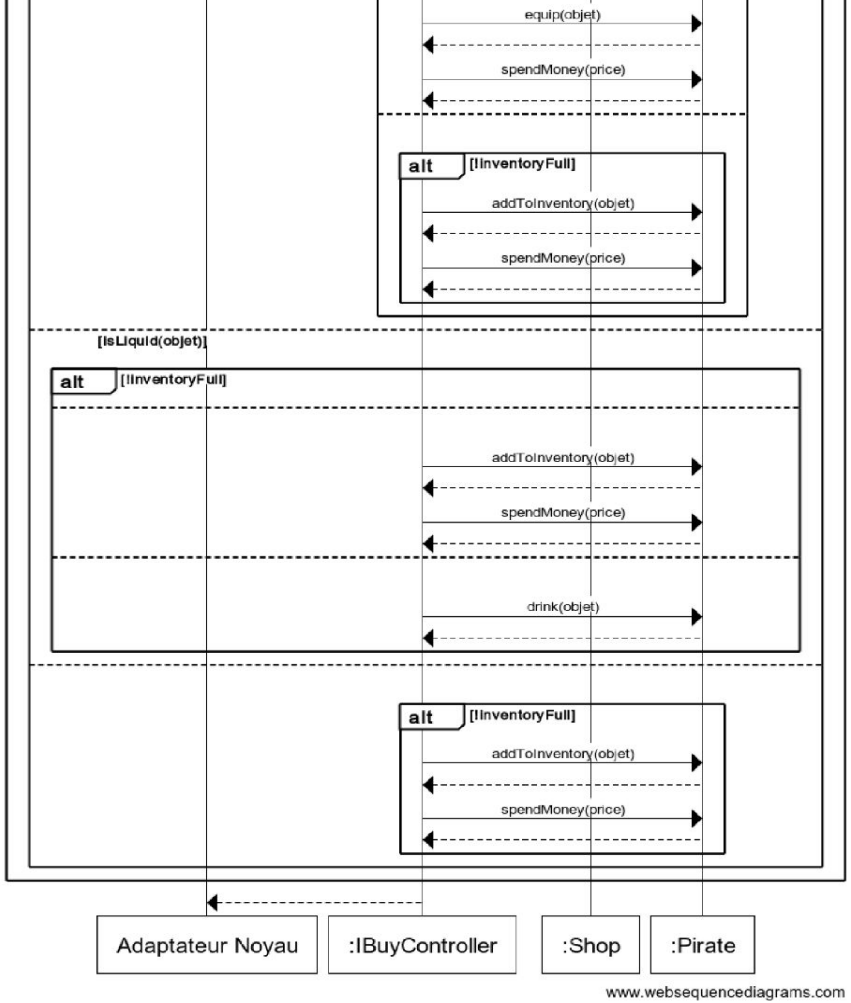



UML Pirates Class Diagram
Market
Operations
+removeItem(UseableItem item)
+getItemsForSale : UseableItem[]
IntoxicationGauge
Attributes
-level: float
Operations
+getLevel(): float
+setLevel(newLevel: float)
Inventory
Attributes
-maxCapacity: integer
Operations
+add(item: Item)
+insert(item: Item, index: integer)
+remove(item: Item)
+get(index: integer)
+contains(item: Item): boolean
Pirate
Attributes
-name: string
-coins: integer
-healthPoint: integer
-maxHealthPoint: integer
-weapon: Weapon
-inventory: Inventory
-intoxicationGauge: IntoxicationGauge
Operations
+getString(): string
+getCoins(): integer
+setCoins(coins: integer)
+getHealthPoints(): integer
+setHealth(health: integer)
+getMaxHealth(): integer
+getEquippedWeapon(): Weapon
+equipWeapon(weapon: Weapon)
+give(Item item)
+drink(Liquid liquid)
+getInventory(): Inventory
+getIntoxicationGauge(): IntoxicationGauge
+getBoard(): Board
+setBoard(board: Board)
<<abstract>>
Item
Attributes
-namespace: string
Operations
+getOwner(): Pirate
+setOwner(newOwner: Pirate)
+getNamespace(): string
+drop()
<<abstract>>
UseableItem
Attributes
-price: int
Operations
<<abstract>> +use()
+getprice: int
+setprice:int
<<abstract>>
Weapon
Attributes
-fightBonus: float
-damages: integer
-stealingPotential: float
Operations
+getFightBonus(): float
+getDamages(): integer
+getStealingPotential(): float
+use()
<<abstract>>
Liquid
Attributes
-recovery: integer
-intoxication: float
Operations
+use()
Board
Attributes
-merchant: int
-boardSize: int
Operations
+getCell(cellNumber: integer): Cell
+getBoardSize(): integer
+getMerchantCellNumber(): integer
+getPirateLocation(pirate: Pirate): integer
+addPirate(pirate: Pirate)
+movePirateTo(pirate: Pirate, cellNumber: integer)
+canMoveToMerchant(cellNumber: integer): boolean
Cell
Attributes
-num: int
Operations
<<abstract>> +getType(): CellType
+getNum(): integer
+getDroppedItems(): List<Item>
Chest
Attributes
-coins: integer
-item: Item
+getItem(): Item
+getCoins(): integer
Trap
Attributes
-trapType: TrapType
Operations
+getTrapType(): TrapType
+getType(): CellType
Land
Operations
+getType(): CellType
Store
Operations
+getType(): CellType
BoardFactory
Attributes
-numberOfCells: integer
-merchantPossiblesCells: integer[1..*]
-specialCellOdds: float
-chestOdds: float
-chestGoldBounds: integer[2]
Operations
<<static>> +getFactory(): BoardFactory
<<static>> +getDefaultBoard(): Board
+setNumberOfCells(numberOfCells: integer)
+setMerchantPossibleCells(merchantPossibleCells: integer[])
+setSpecialCaseOdds(specialCaseOdds: float)
+setChestOdds(chestOdds: float)
+setChestGoldBounds(chestGoldBounds: integer[])
+build(): Board
Class
Abstract class
Interface
Enumeration
Association
Inheritance
Implementation
Dependency
Composition

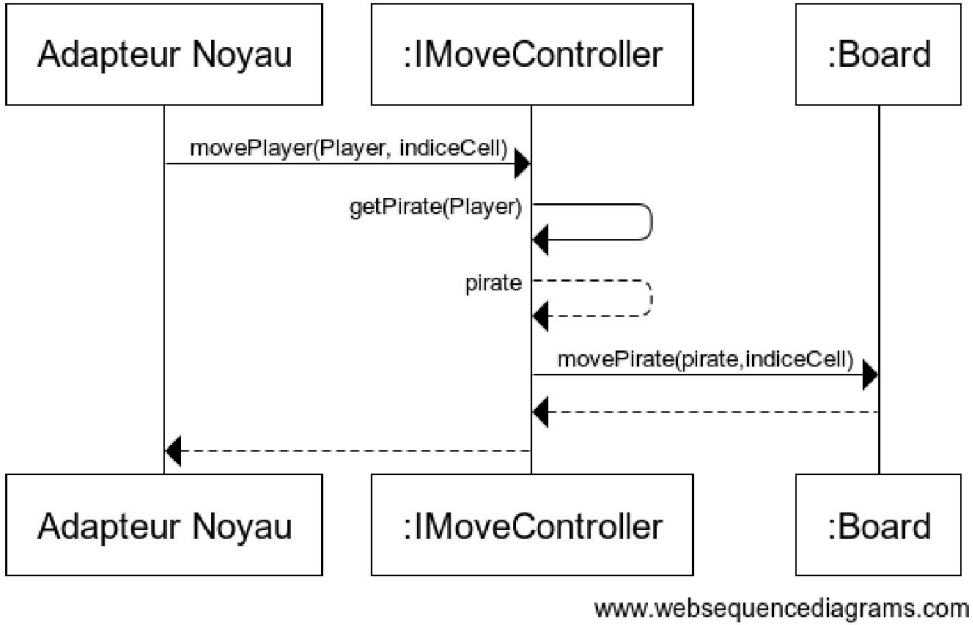
UseCase


UseCase
User
take an item
move a player
examine an item
buy an item
drop an item
<<extends>>
<<extends>>
Equip with a weapon
drink a liquid


Call Of Rhum_Diagramme de séquence 1
Adaptateur Noyau
:IBuyController
:Shop
:Pirate
buy(indiceObjet)
getPrice(indiceObjet)
price
getSavings()
savings
alt
[savings>=price]
getItem(indiceObjet)
objet
isInventoryFull()
inventoryFull
alt
[isWeapon(objet)]
getEquipedWeapon()
weapon
alt
[weapon==NULL]
equip(objet)
spendMoney(price)
alt
[!inventoryFull]
addToInventory(objet)
spendMoney(price)
[isLiquid(objet)]


equip(objet)
spendMoney(price)
alt
[!InventoryFull]
addToInventory(objet)
spendMoney(price)
[isLiquid(objet)]
alt
[!InventoryFull]
addToInventory(objet)
spendMoney(price)
drink(objet)
alt
[!InventoryFull]
addToInventory(objet)
spendMoney(price)
Adaptateur Noyau
:IBuyController
:Shop
:Pirate
www.websequencediagrams.com

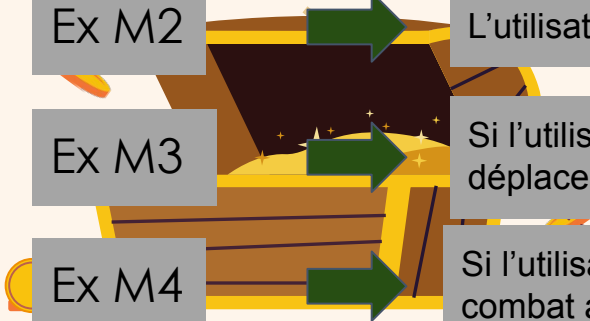
Call Of Rhum_Diagramme de séquence 2



Spécification détaillée

Cas Déplacer un joueur

Acteur : Utilisateur

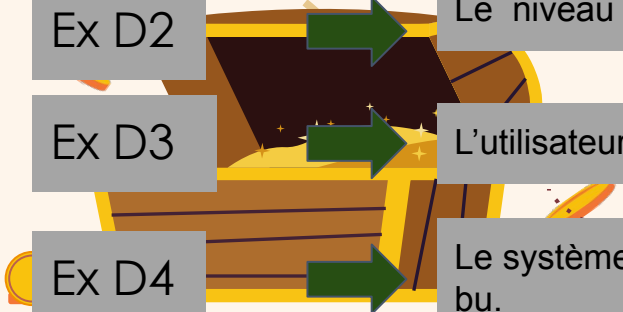


Ex M1	L'utilisateur ayant fait un déplacement sur la dernière case, est le vainqueur du jeu.
Ex M2	L'utilisateur doit se déplacer d'un nombre de cases comprises entre 2 et 12.
Ex M3	Si l'utilisateur doit se déplacer au-delà de la dernière case, il reprend son déplacement à partir de la première case.
Ex M4	Si l'utilisateur se déplace sur une case où il y a un autre utilisateur, alors un combat aura lieu entre les deux utilisateurs.
Ex M5	L'utilisateur ayant fait un déplacement sur une case spéciale déclenche l'effet de la case spéciale.

Spécification détaillée

Cas Boire un liquide

Acteur : Utilisateur



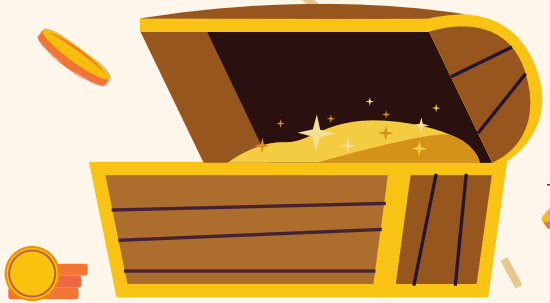
Ex D1	Le système doit vérifier que l'utilisateur possède le liquide.
Ex D2	Le niveau d'intoxication de l'utilisateur varie selon le liquide bu.
Ex D3	L'utilisateur a la possibilité de boire un liquide.
Ex D4	Le système doit enlever le liquide dans l'inventaire de l'utilisateur après avoir été bu.
Ex D5	Le niveau de vie de l'utilisateur varie selon le liquide bu.

Lambdas

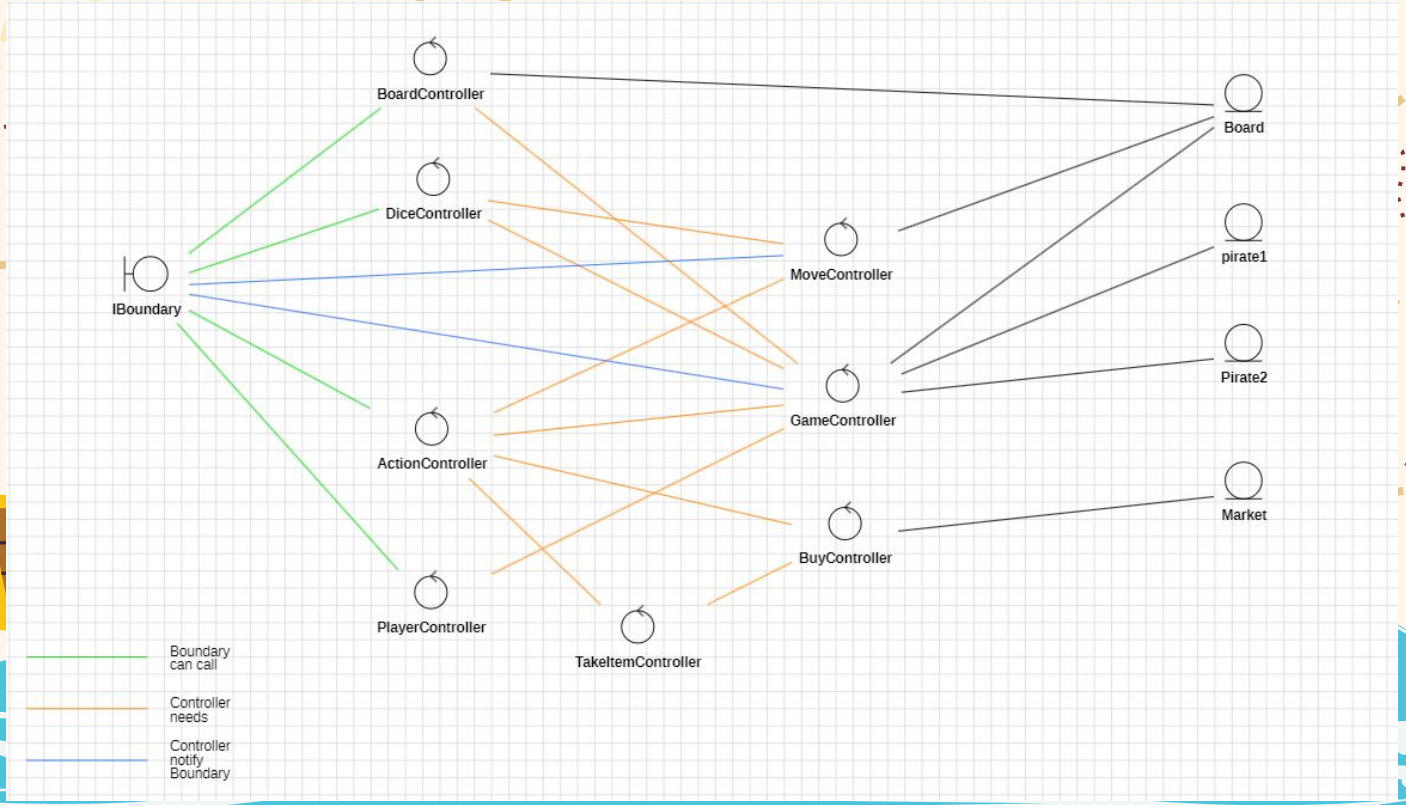
```
private Optional<Integer> findFirst(Item item) {
    Predicate<Item> criteria;
    Item current;
    int i = 0;
    while (i < maxCapacity) {
        current = items[i];
        criteria = current == null || item == null ? it -> it == item : it -> it.equals(item);
        if (criteria.test(current))
            return Optional.of(i);
        i++;
    }
    return Optional.empty();
}
```

```
private void animDice(Dice dice){
    Runnable myThread = () ->
    {
        for (int i =0; i < 6; i++ ){
            dice.setFaceValue(generateRandomNumber());
            wait(90);
        }

    };

    Thread t = new Thread(myThread);
    t.start();
}
```

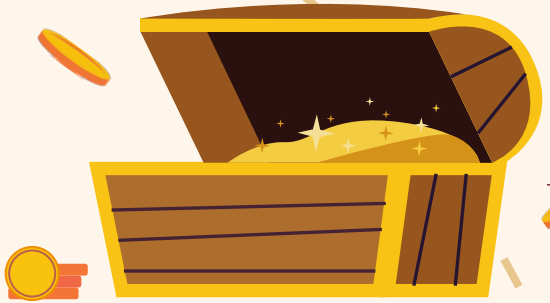
Architecture



Architecture



Code - apport des lambdas



Test Fonctionnel

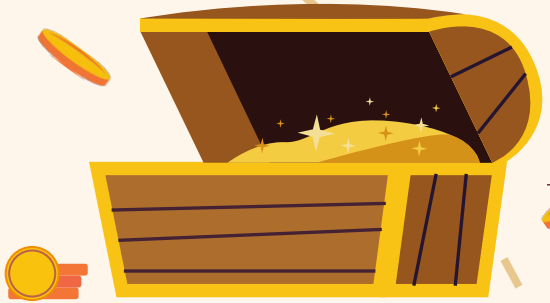
InventoryTest.java

```java
    loferga
    @Test
    public void addTest() {
        inv.add(item1);
        inv.add(item2);
        assertEquals(item1, inv.get(0));
        assertEquals(item2, inv.get(1));
        inv.add(item1);
        assertThrows(NoFreeSlotException.class, () -> inv.add(item1));
    }

    loferga
    @Test
    public void insertTest() {
        inv.insert(item1, index: 2);
        inv.insert(item2, index: 1);
        assertEquals(item1, inv.get(2));
        assertEquals(item2, inv.get(1));
    }

    loferga
    @Test
    public void removeTest() {
        inv.add(item1);
        inv.add(item2);
        inv.remove(item1);
        assertNull(inv.get(0));
        assertEquals(item2, inv.get(1));
    }
```

InventoryTest.java | BoardFactoryTest.java | BoardTest.java

```java
    loferga
    @Before
    public void setUp() throws Exception {
        pirate = new Pirate( name: "pirate", coins: 0, maxHealthPoints: 5);
        otherPirate = new Pirate( name: "other pirate", coins: 0, maxHealthPoints: 5);
        board = BoardFactory.getDefaultBoard(new ItemRegistry() /* empty registry */);
        board.addPirate(pirate);
        board.addPirate(otherPirate);
    }

    loferga
    @Test
    public void movePirateTest() {
        int previousLocation;
        // test move pirate to cell number
        board.movePirateTo(pirate, cellNumber: 4);
        assertEquals( expected: 4, board.getCell(pirate).getNum());
        // test edge case with big values
        previousLocation = board.getPirateLocation(pirate);
        board.movePirateTo(pirate, cellNumber: 10000);
        assertEquals(previousLocation, board.getPirateLocation(pirate));
        // test edge case with no movement
        previousLocation = board.getPirateLocation(pirate);
        board.movePirateTo(pirate, board.getPirateLocation(pirate));
        assertEquals(previousLocation, board.getPirateLocation(pirate));
        // test move pirate to pirate
        board.movePirateTo(pirate, otherPirate);
        assertEquals(board.getPirateLocation(pirate), board.getPirateLocation(otherPirate));
    }

    1 usage  loferga
```

Code - démonstration



Bilan