VIETNAM NATIONAL UNIVERSITY - HO CHI MINH CITY

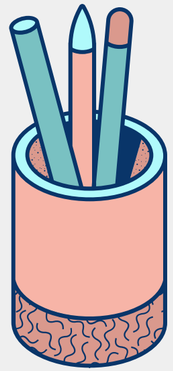
INTERNATIONAL UNIVERSITY

OBJECT-ORIENTED PROGRAMMING PROJECT

# Platform Game "Mysterious Journey"

TEAM 19

# Team Members

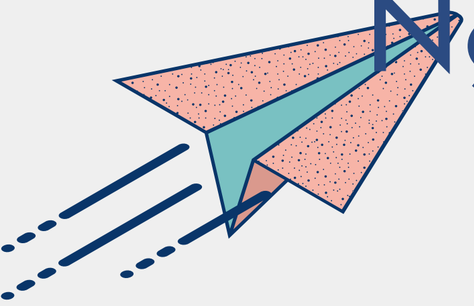


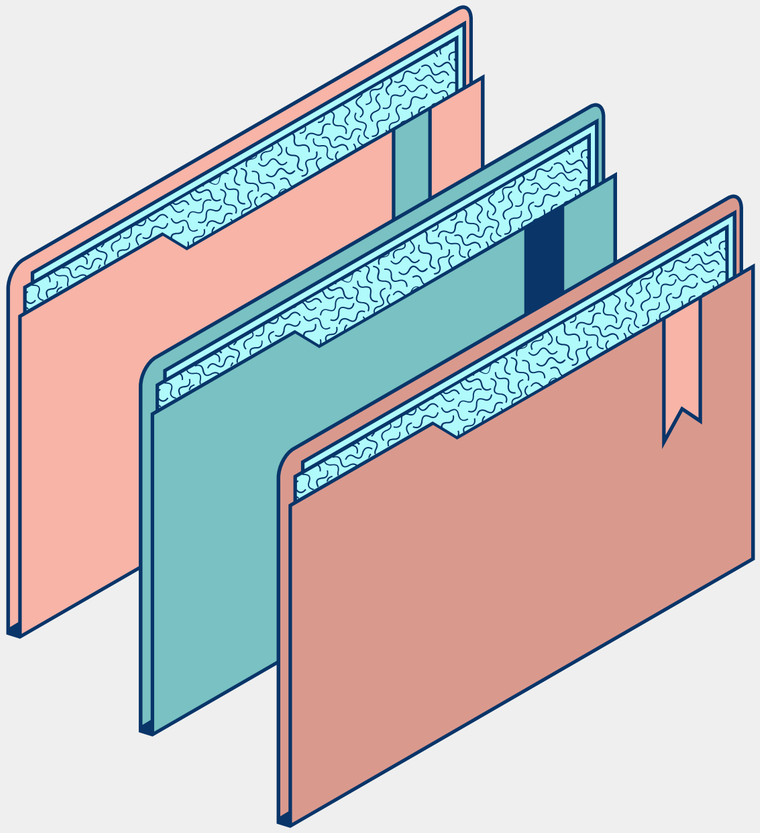
Nguyễn Phúc Đạt - ITITIU22028

Trần Lưu Hồng Phương - ITITIU22198

Nguyễn Hoàng Thảo Trinh - ITCSIU22

Nguyễn Thành Nam - ITCSIU22311



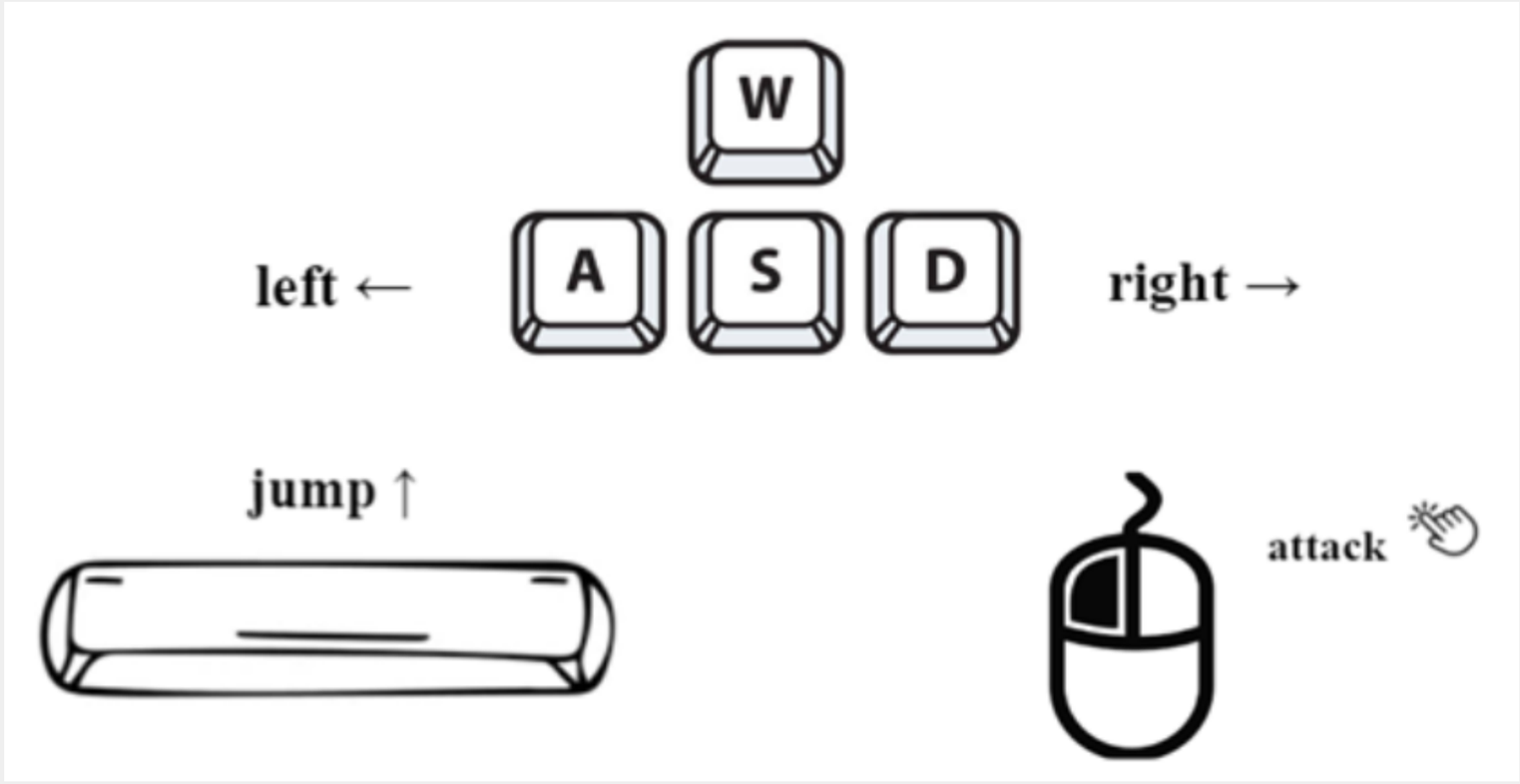


# Table of Content

# I. Introduction

- "Mysterious Journey"
- A platform game with controlled jumping
- Basic concept of OOP

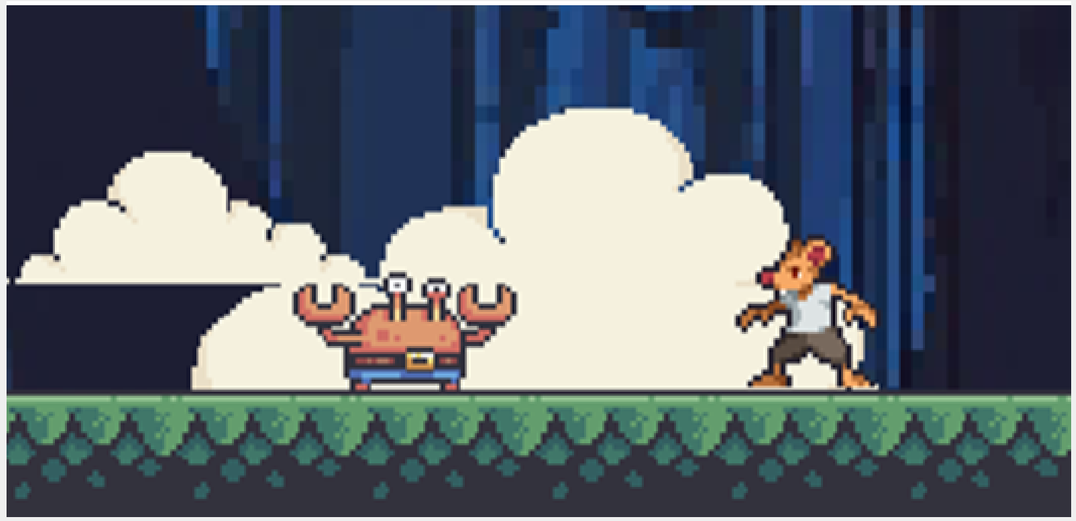
# II. Game Rules



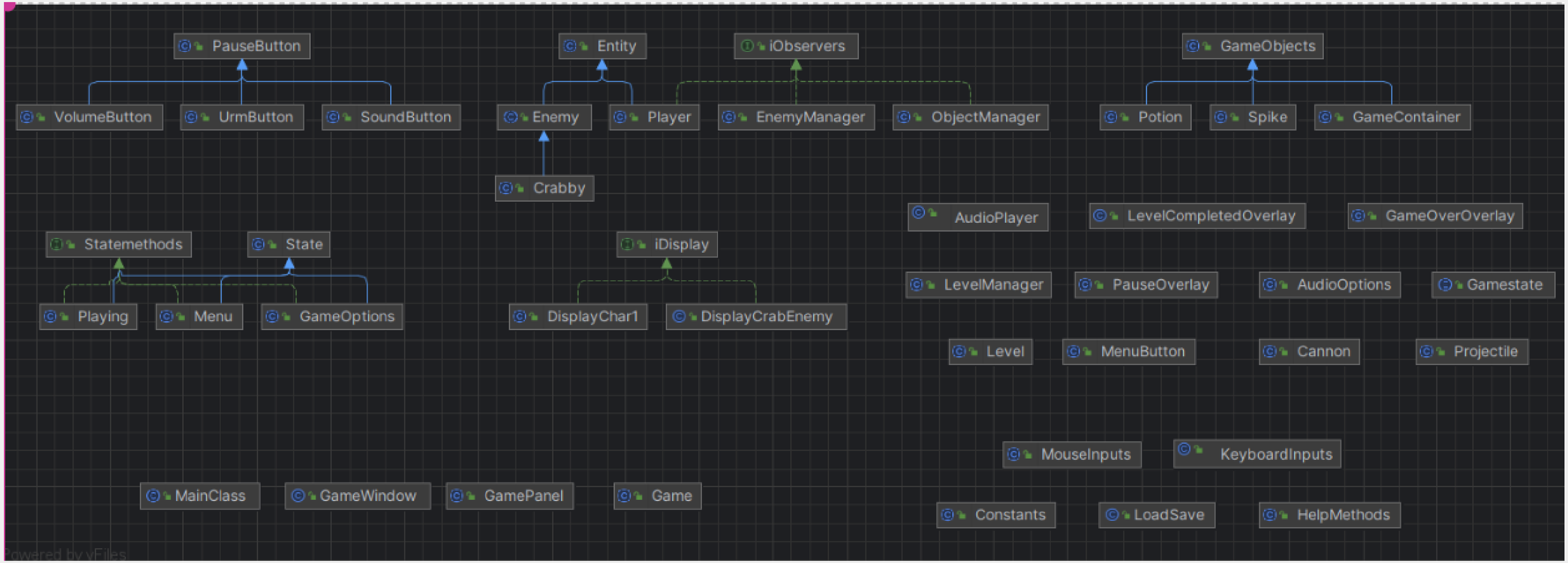

- "A" keypress: move left
- "D": move right
- "space": jump
- left-side mouse: attack
- right-side mouse: power-attack

# II. Game Rules

- kill all enemies
- avoid traps and canon attack
- collect items to upgrade abilities



# III. Class Diagram



# IV. Applied Design Patterns

## 1. Singleton

```
NP-Dat +3
public class Game implements Runnable {
    3 usages
    private static Game instance = null; // singleton
```

```
2 usages   thaortrinh
public static Game getInstance() {
    if (instance == null) {
        instance = new Game();
    }
    return instance;
}
```

```
1 usage   NP-Dat +2
private Game() {
    initClasses();

    gamePanel = new GamePanel(this);
    gameWindow = new GameWindow(gamePanel);
    gamePanel.setFocusable(true);
    gamePanel.requestFocus();

    startGameLoop();
}
```

```
NP-Dat +1 *
public class MainClass {
    NP-Dat +1 *
    public static void main(String[] args) {

        Game platformGame = Game.getInstanc
    }

}
```

# IV. Applied Design Patterns

## 2. Observer

```
package gamestates;

import javax.security.auth.Subject;

3 implementations  NP-Dat
public interface iObservers {

    3 implementations  NP-Dat
    void resetAll();

}
```

Choose Implementation of iObservers (3 found)

- EnemyManager (entities) — PD_Test1
- ObjectManager (objects) — PD_Test1
- Player (entities) — PD_Test1

```
NP-Dat +2
public void resetAll() {
    gameOver = false;
    paused = false;
    levelCompleted = false;
    playerDying = false; // player alive in new game after die

    for (iObservers observer: observers){
        observer.resetAll();
    }
}
```

```
NP-Dat +3
private void initClasses() {
    levelManager = new LevelManager(game);
    enemyManager = new EnemyManager( playing: this);
    objectManager = new ObjectManager( playing: this);
    player = new Player( x: 200,  y: 200, (int) (64 * Game.SCALE), (int) (40 * Game.SCALE),  playing: this);
    player.loadLvlData(levelManager.getCurrentLevel().getLevelData());
    player.setSpawn(levelManager.getCurrentLevel().getPlayerSpawn());
    pauseOverlay = new PauseOverlay( playing: this);
    gameOverOverlay = new GameOverOverlay( playing: this);
    levelCompletedOverlay = new LevelCompletedOverlay( playing: this);

    observers.add(objectManager);
    observers.add(player);
    observers.add(enemyManager);
```

# IV. Applied Design Patterns

## 3. Strategy

```
2 implementations   NP-Dat
public interface iDisplay {
    2 implementations   NP-Dat
    BufferedImage[][] loadAnimations();
}
```

Choose Implementation of iDisplay (2 found)

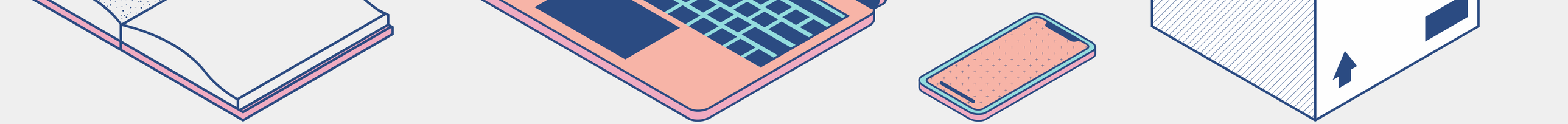
- DisplayChar1 (entities.display) — PD_Test1
- DisplayCrabEnemy (entities.display) — PD_Test1

```
NP-Dat
private void loadEnemyImgs() {
    iDisplay displayCrabEnemy = new DisplayCrabEnemy();
    crabbyArr = displayCrabEnemy.loadAnimations();
}
```

```
NP-Dat
private void loadAnimations() {
    iDisplay displayChar1 = new DisplayChar1();
    this.animations = displayChar1.loadAnimations();

    statusBarImg = LoadSave.GetSpriteAtlas(LoadSave.STATUS_BAR);

}
```



# V. Conclusion

1 — 2 — 3

## SUMMARY

**Implementation and application**

SOLID principle
Design patterns

## LIMITATION

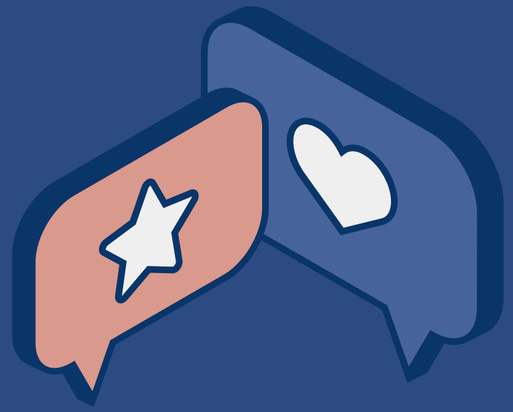
**Coding experience**

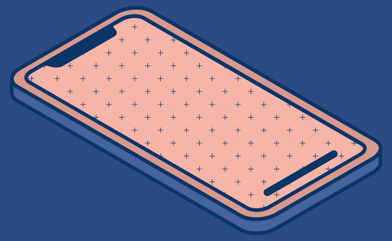
Unfamiliarity with game dev
Limited coding experience

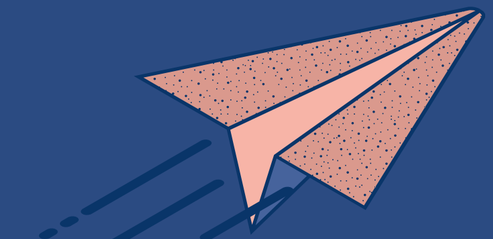
## EVALUATION

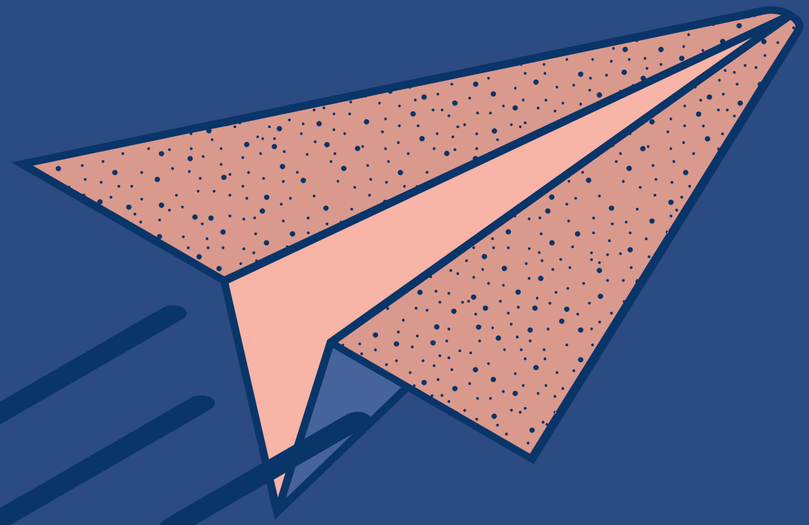
**Expansion & Upgradation**

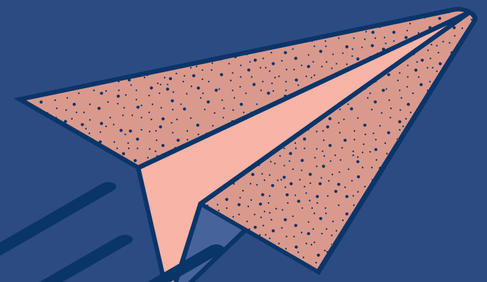
Code designed to be expanded

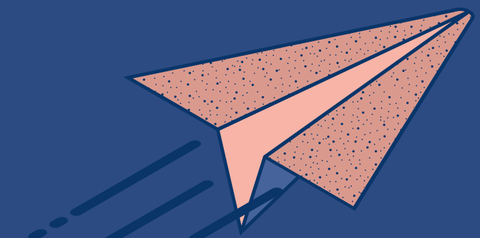












# Thanks

## for your attention!