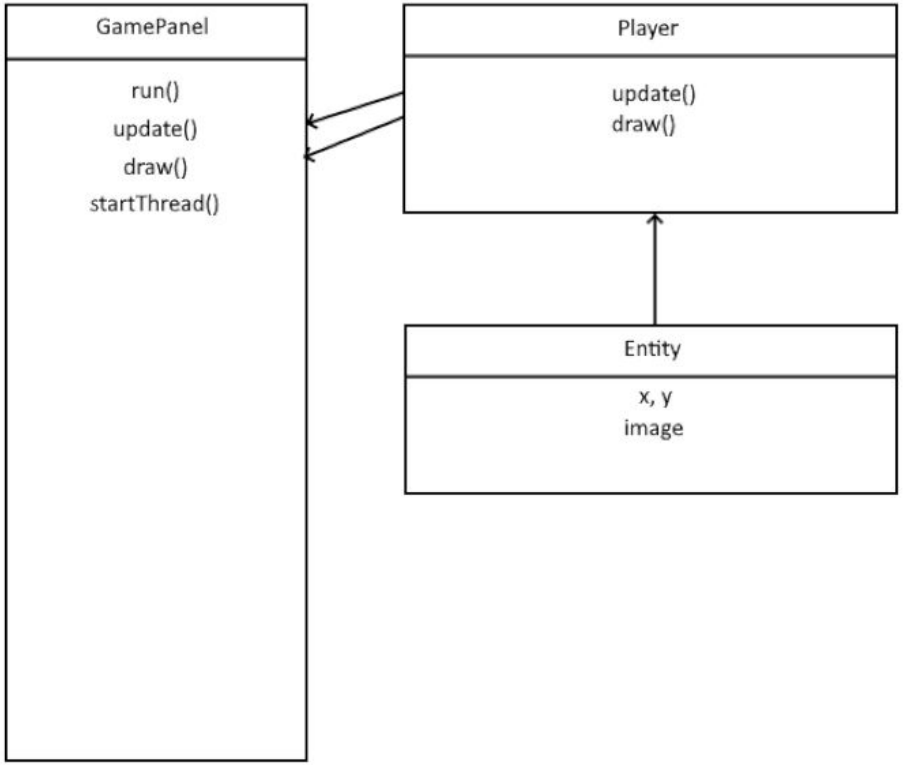
Класи Entity, Player, спрайти та анімації

План на лекцію

- Персонажі
- Клас Entity
- Клас Player
- Використання спрайтів
- Папка для ресурсів
- Як використовувати спрайти
- Анімація
- Стан спокою
- Результати

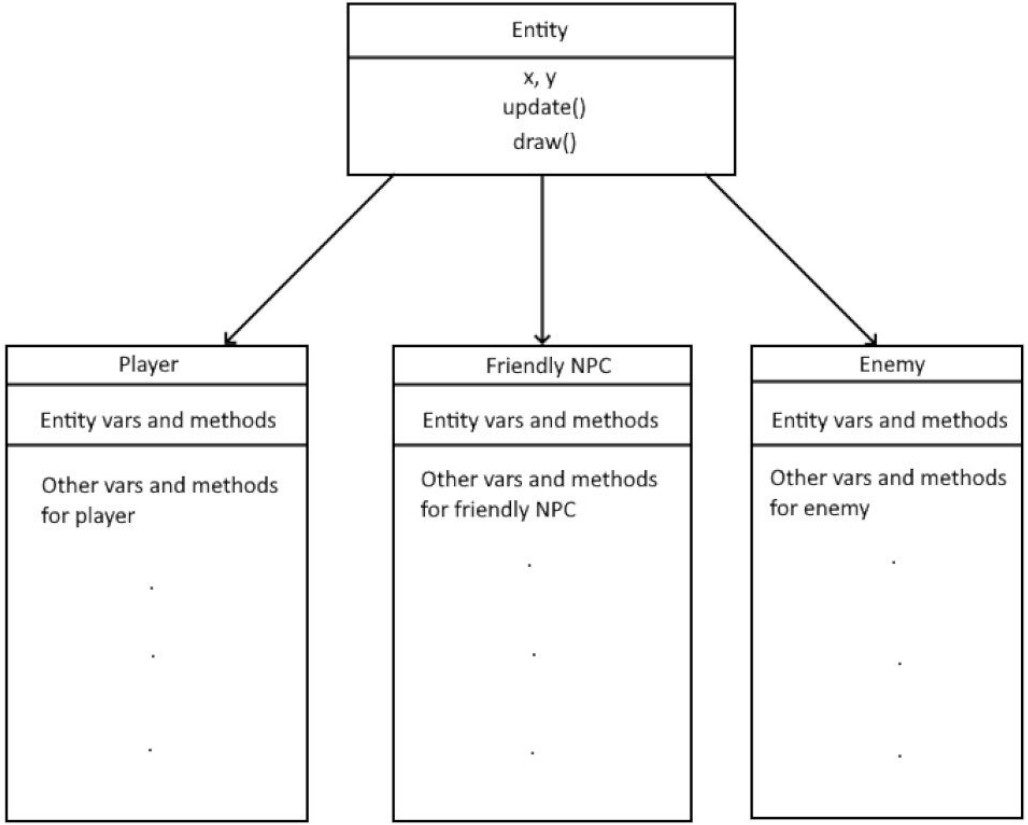
Персонажі



Зазвичай для обробки даних персонажів та інших створінь їх дані виносяться в окремий клас.

Це зменшує кількість коду в основному циклі та допомагає забезпечити ієрархію усіх створінь на ігровій сцені.

Клас Entity


Entity
x, y
update()
draw()
Player
Entity vars and methods
Other vars and methods for player
Friendly NPC
Entity vars and methods
Other vars and methods for friendly NPC
Enemy
Entity vars and methods
Other vars and methods for enemy

Клас Entity

```
package entities;

import java.awt.Graphics2D;
import java.awt.image.BufferedImage;

public abstract class Entity {
    public int x, y;
    public int speed;

    public BufferedImage up1,up2,down1,down2,left1,left2,right1,right2;
    public BufferedImage standUp, standDown, standLeft, standRight;
    public String direction;

    public int spriteCounter = 0, spriteNum = 1;

    public abstract void update();
    public abstract void draw(Graphics2D g2);
}
```

- створити змінні x, y, для відображення положення створіння, та speed, для відображення зміни позиції створіння
- створити змінні, що зберігають спрайти створінь
- створити змінні для позначення напряму персонажа, а також рахунку спрайту
- створити абстрактні методи для подальшої реалізації

Клас Player

Цей клас зберігає усі дані, що пов’язані з основним персонажем, який керується гравцем. Він наслідує клас Entity, тому має усі його змінні та методи. Для Player методи update() та draw() перевизначаються. Також додається пара методів, що дозволяють зробити початкове налаштування ігрового персонажа.

Екземпляр цього класу використовується для того, щоб виконувати оновлення усієї отриманої інформації з клавіатури, а також робити відображення персонажа на ігровій сцені.

Клас Player

Для правильної роботи класу, підключаємо ігрову панель та слухач клавіатури. Далі створюємо метод setDefaultValues, що виставляє стандартні початкові значення, а також getPlayerImage, що завантажує спрайти до змінних класу Entity.

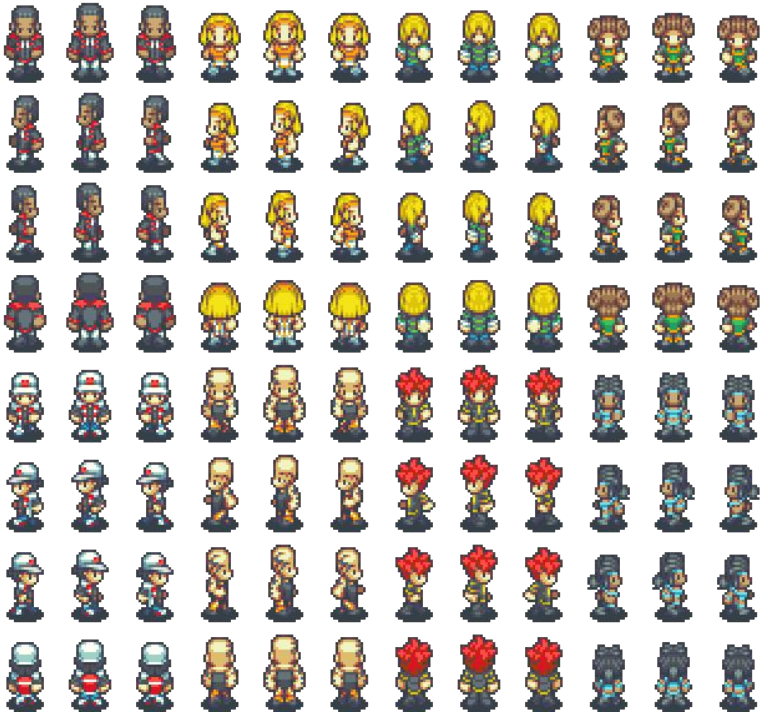
```
public class Player extends Entity {
    GamePanel gp;
    KeyHandler keyHandler;

    public Player(GamePanel gp,
    KeyHandler keyHandler) {
        this.gp = gp; this.keyHandler = keyHandler;
        setDefaultValues(); getPlayerImage();
    }
    public void setDefaultValues() {
        x = 100; y = 100;
        speed = 4;
        direction = "down";
    }
    public void getPlayerImage() {}
    @Override
    public void update() {}

    @Override
    public void draw(Graphics2D g2) {}


}
```

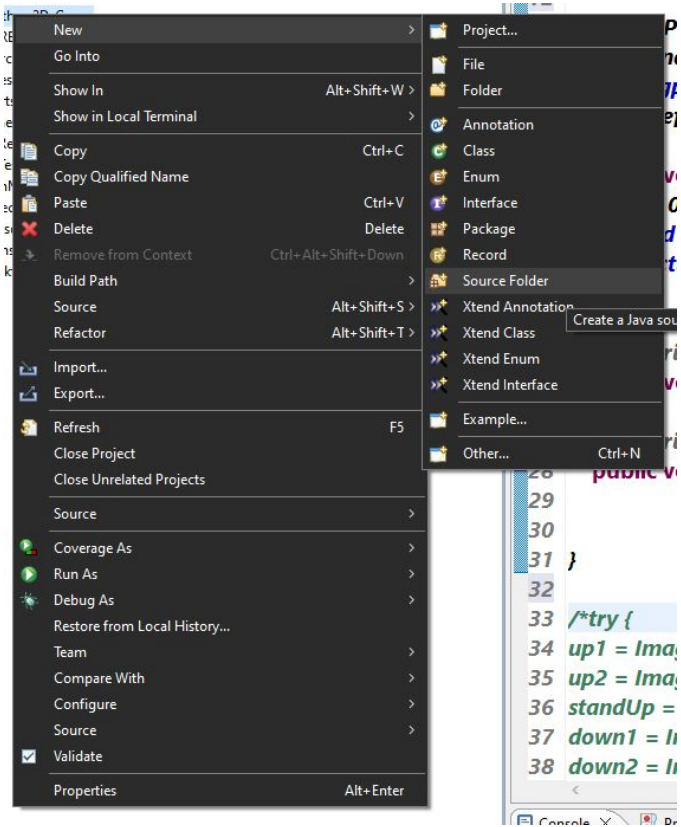
Використання спрайтів

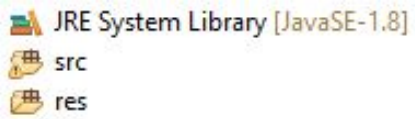


Спрайт - двовимірне растрове зображення, що використовується в 2D-іграх. Може бути окремим для кожної частини об’єкту, складати повністю весь об’єкт, або закривати декілька об’єктів однією картинкою.

Спрайти дуже популярні при розробці простих 2D-ігор в інді-жанрі.

res - папка з ресурсами





Папка res слугує основним містилищем усіх додаткових файлів, що не є кодом, але доповнюють чи навіть складають основну частину програми.

Створюється за допомогою меню Source Folder при натисканні ПКМ на кореневу папку проекту.

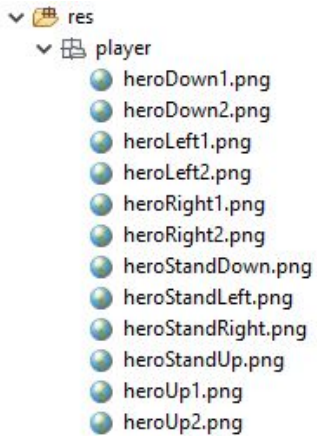
BufferedImage

Даний клас використовується для представлення зображень у коді та подальшого їх використання у програмі.

В даному випадку він зберігає зображення всіх спрайтів для відображення їх на ігровій сцені.

Оскільки кожний ігровий об’єкт має бути відображений на сцені, то даний клас буде встановлений в усі тайли та створінь.

Завантаження зображень




```
public void getPlayerImage() {
  try {
    up1 = ImageIO.read(getClass().getResourceAsStream("/player/heroUp1.png"));
    up2 = ImageIO.read(getClass().getResourceAsStream("/player/heroUp2.png"));
    standUp = ImageIO.read(getClass().getResourceAsStream("/player/heroStandUp.png"));
    down1 = ImageIO.read(getClass().getResourceAsStream("/player/heroDown1.png"));
    down2 = ImageIO.read(getClass().getResourceAsStream("/player/heroDown2.png"));
    standDown = ImageIO.read(getClass().getResourceAsStream("/player/heroStandDown.png"));
    left1 = ImageIO.read(getClass().getResourceAsStream("/player/heroLeft1.png"));
    left2 = ImageIO.read(getClass().getResourceAsStream("/player/heroLeft2.png"));
    standLeft = ImageIO.read(getClass().getResourceAsStream("/player/heroStandLeft.png"));
    right1 = ImageIO.read(getClass().getResourceAsStream("/player/heroRight1.png"));
    right2 = ImageIO.read(getClass().getResourceAsStream("/player/heroRight2.png"));
    standRight = ImageIO.read(getClass().getResourceAsStream("/player/heroStandRight.png"));
    } catch (IOException e) {
    e.printStackTrace();
    }
}
```

- в папці res створюємо папку player, куди копіюємо усі спрайти
- у методі getPlayerImage() завантажуємо скопійовані спрайти до змінних BufferedImage

Оновлення даних

```
@Override
public void update() {
    if(keyHandler.up) {y-=speed;direction = "up";}
    if(keyHandler.down) {y+=speed;direction = "down";}
    if(keyHandler.left) {x-=speed;direction = "left";}
    if(keyHandler.right) {x+=speed;direction = "right";}

    spriteCounter++;
    if(spriteCounter > 12) {
        if(!keyHandler.up&&
            !keyHandler.down&&
            !keyHandler.left&&
            !keyHandler.right) spriteNum = 0;
        else if(spriteNum == 1) spriteNum++;
        else if(spriteNum == 2 || spriteNum == 0) spriteNum = 1;
        spriteCounter = 0;
    }
}
```

У методі update() робимо читання станів клавіатури.
Кожен стан відповідає за переміщення персонажа по ігровому полю.
Також стани змінюють напрям персонажа.
Нижче додано лічильник кадрів.
Після 12 оновлень настає час змінювати спрайт.
У випадку, коли персонаж стоїть на місці, персонаж залишає спрайт “АФК”.

Малювання анімації

```
@Override
public void draw(Graphics2D g2) {
    BufferedImage img = null;
    switch (direction) {
    case "up":
        if(spriteNum == 1) img = up1;
        else if(spriteNum == 2) img = up2;
        else if(spriteNum == 0) img = standUp;
        break;
    case "down":
        if(spriteNum == 1) img = down1;
        else if(spriteNum == 2) img = down2;
        else if(spriteNum == 0) img = standDown;
        break;
    case "left":
        if(spriteNum == 1) img = left1;
        else if(spriteNum == 2) img = left2;
        else if(spriteNum == 0) img = standLeft;
        break;
    case "right":
        if(spriteNum == 1) img = right1;
        else if(spriteNum == 2) img = right2;
        else if(spriteNum == 0) img = standRight;
        break;
    default:
        break;
    }
    g2.drawImage(img, x, y, gp.tile, gp.tile, null);
}
```

У створеному методі draw() додано Додаткову змінну BufferedImage, що тримає поточне зображення персонажа. В залежності від напряму, стану спокою або переміщення, до цієї змінної подаються інші зображення, що зберігаються у класі Entity.

Коли метод вирішить, який саме спрайт завантажити, метод drawImage() завантажує його відносно координат персонажа, за розміром тайла.

Перетворення панелі

```
Player player = new Player(this, kH);

public void update() {
    player.update();
}

@Override
protected void paintComponent(Graphics g) {
    super.paintComponent(g);
    Graphics2D g2 = (Graphics2D)g;
    player.draw(g2);
    g2.dispose();
    g.dispose();
}
```

Останніми кроками для отримання результату буде декларація змінної персонажа на ігровій панелі. Усі змінні x, y, speed звідси видаляються, оскільки тепер використовуються їх аналоги у класі Player.

Методи update() та paintComponent() викликають однойменні методи персонажа, змінюючи дані та картинку.

Перший результат