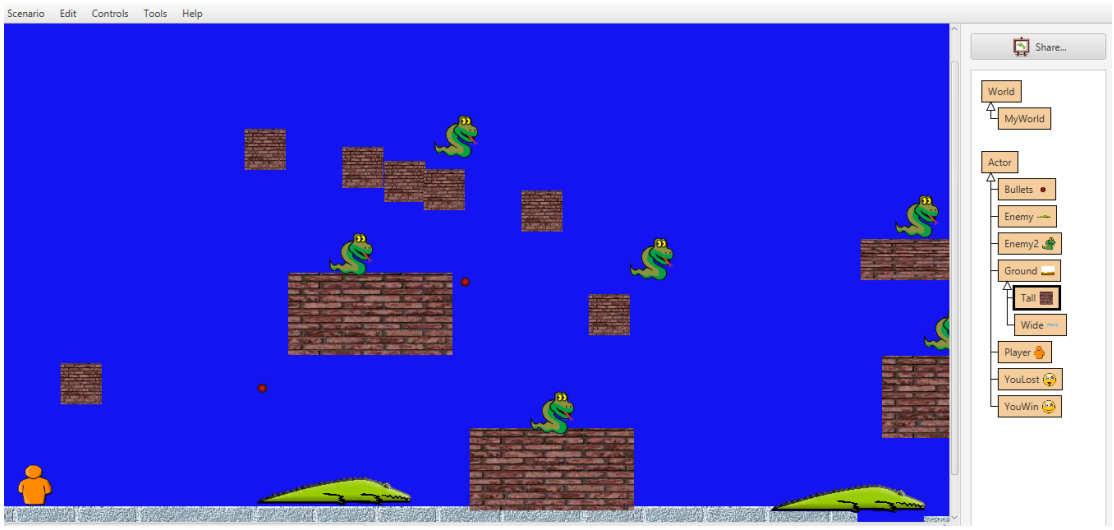
Άσκηση 9η

Δημιουργήστε τις κλάσεις και τις υποκλάσεις όπως απεικονίζονται παρακάτω, γράψτε τους κώδικες στους αντίστοιχους editor και «τρέξτε» το πρόγραμμα….



Το παρακάτω πρόγραμμα είναι γραμμένο σε Java και χρησιμοποιεί το framework Greenfoot για τη δημιουργία ενός παιχνιδιού.

MyWorld class:

Η κλάση επεκτείνει την World και δημιουργεί έναν κόσμο διαστάσεων 3000x700 pixel, με έναν παίκτη (Player) και ορισμένα άλλα αντικείμενα.

Η μέθοδος prepare() προσθέτει αντικείμενα τύπου Tall, Wide, Enemy, και Enemy2 στον κόσμο.

Tall class:

Η κλάση περιγράφει ένα αντικείμενο τύπου "Tall" (ύψος μεγαλύτερο από το πλάτος).

Wide class:

Η κλάση περιγράφει ένα αντικείμενο τύπου "Wide" (πλάτος μεγαλύτερο από το ύψος).

Player class:

Δεν παρέχεται ακόμα κώδικας για την κλάση Player.

Enemy class:

Η κλάση περιγράφει έναν εχθρό.

Enemy2 class:

Η κλάση περιγράφει έναν δεύτερο τύπο εχθρού.

```
import greenfoot.*;
/**
 * Kaffetzis Vasileios
 * 15/11/2023
 */
public class MyWorld extends World
{
    public MyWorld()
    // Δημιουργία νέου κόσμου διαστάσεων 3000x700 pixels με μέγεθος κελιών 1x1 pixel.
    {
        // Create a new world with 600x400 cells with a cell size of 1x1 pixels.
        super(3000, 700, 1);
        // Ορισμός χρώματος φόντου του κόσμου σε μπλε.
        getBackground().setColor(new Color (20,20,240));
        // Συμπλήρωση του φόντου με το καθορισμένο χρώμα.
        getBackground().fill();
        // Προσθήκη τριών αντικειμένων τύπου Wide σε συγκεκριμένες θέσεις.
        addObject(new Wide(), 85, 666);
        addObject(new Wide(), 450, 666);
        addObject(new Wide(), 850, 666);
        // Προσθήκη ενός αντικειμένου τύπου Player σε συγκεκριμένη θέση.
        addObject(new Player(),50, 500);
        // Εκτέλεση της μεθόδου prepare() για περαιτέρω προετοιμασία του κόσμου.
        prepare();
    }
    private void prepare()
    {
        // Δημιουργία αντικειμένου τύπου Tall με συγκεκριμένες διαστάσεις.
        Tall tall = new Tall(200, 100);
        addObject(tall,1014,486);       // Προσθήκη του tall σε συγκεκριμένη θέση.
        Tall tall2 = new Tall(200, 100);
        addObject(tall2,1287,354);
        tall2.setLocation(1063,253);
        Tall tall3 = new Tall(100, 100);
        addObject(tall3,1361,201);
        tall2.setLocation(925,204);
        Tall tall4 = new Tall(200, 100);
        addObject(tall4,1135,383);
        tall4.setLocation(628,349);
        Tall tall5 = new Tall(50, 200);
        addObject(tall5,1165,382);
        tall5.setLocation(1252,428);
        Tall tall6 = new Tall(200, 50);
        addObject(tall6,1178,370);
        tall6.setLocation(1023,362);
        Wide wide = new Wide();
        addObject(wide,1542,660);
        wide.setLocation(1486,658);
        Wide wide2 = new Wide();
        addObject(wide2,2196,656);
        Tall tall7 = new Tall(200, 200);
        addObject(tall7,2347,346);
        tall7.setLocation(2489,420);
        removeObject(tall7);              // Αφαίρεση του tall7 από τον κόσμο.
        Tall tall17 = new Tall(60, 50);
        Enemy2 enemy213 = new Enemy2();
        addObject(enemy213,2840,587);
        enemy213.setLocation(2837,603);
        Enemy enemy7 = new Enemy();
        addObject(enemy7,2578,361);
        Enemy2 enemy214 = new Enemy2();
        addObject(enemy214,745,326);
        Enemy2 enemy215 = new Enemy2();
        addObject(enemy215,507,177);
    }
}
```

Στην υπόκλαση MyWord, προσθέτουμε τα αντικείμενα χειροκίνητα και κάνουμε Save the World

Η κλάση Ground είναι μια υποκλάση της κλάσης Actor και χρησιμοποιείται για να αντιπροσωπεύσει το έδαφος στον κόσμο του παιχνιδιού. Η act μέθοδος παρέχει τον χώρο για την εκτέλεση κώδικα κατά τη διάρκεια της εκτέλεσης του προγράμματος.

```
import greenfoot.*;
public class Ground extends Actor // Κλάση που αντιπροσωπεύει το έδαφος στον κόσμο.
{
    public void act()   // Μέθοδος που εκτελείται κατά την εκτέλεση του προγράμματος.
    {
        //
    }
}
```

Η κλάση Tall είναι υποκλάση της κλάσης Ground και αντιπροσωπεύει ένα ψηλό αντικείμενο στον κόσμο του παιχνιδιού. Ο κατασκευαστής της τίθεται να αλλάξει το μέγεθος της εικόνας του ψηλού αντικειμένου σύμφωνα με τις παρεχόμενες διαστάσεις (πλάτος και ύψος).

```
import greenfoot.*;
public class Tall extends Ground // Κλάση που αντιπροσωπεύει ένα ψηλό αντικείμενο (Tall) στον κόσμο.
{
    // Κατασκευαστής που ορίζει το πλάτος (width) και το ύψος (height) του ψηλού αντικειμένου.
    public Tall (int wight, int height )
    {
        super();        // Κλήση του κατασκευαστή της γονικής κλάσης (Ground).
        getImage().scale(wight, height);  // Αλλαγή του μεγέθους της εικόνας του ψηλού αντικειμένου σύμφωνα με τα δοθέντα πλάτος και ύψος.
    }
    public void act()// Μέθοδος που εκτελείται κατά την εκτέλεση του προγράμματος.
    {
        //
    }
}
```

Η κλάση Wide είναι υποκλάση της κλάσης Ground και αντιπροσωπεύει ένα ευρύ αντικείμενο στον κόσμο του παιχνιδιού. Η μέθοδος act() είναι κενή, αλλά μπορείτε να προσθέσετε κώδικα σε αυτήν για τις ενέργειες που θέλετε να εκτελούνται συνεχώς.

```
import greenfoot.*;
public class Wide extends Ground    // Κλάση που αντιπροσωπεύει ένα ευρύ αντικείμενο (Wide) στον κόσμο.
{
    public void act()
    {
        //
    }
}
```

Η παρακάτω κλάση Player αντιπροσωπεύει τον παίκτη στο παιχνίδι και περιλαμβάνει λειτουργίες για την κίνηση, τα άλματα, το πυροβολισμό, και τον έλεγχο συγκρούσεων με εχθρικά αντικείμενα.

```java
import greenfoot.*;
// Κλάση που αντιπροσωπεύει τον παίκτη (Player) στον κόσμο του παιχνιδιού.
public class Player extends Actor
{
    int enemyTouchCount = 0; // Μεταβλητή που καταγράφει τον αριθμό των συγκρούσεων με εχθρικά αντικείμενα.
    int vSpeed = 0; // Κατακόρυφη ταχύτητα για τον προσομοιωμένο πτώση/άλμα.
    int accel = 0; // Επιτάχυνση (δεν χρησιμοποιείται στον κώδικα).
    int speed = 5; // Ταχύτητα κίνησης προς τα δεξιά/αριστερά.
    private int lives = 20; // Αριθμός ζωών του παίκτη.
    private int touch = 0; // Προσωρινή μεταβλητή για ανίχνευση συγκρούσεων.
    boolean checkEnemyCollision = false; // Έλεγχος συγκρούσεων με εχθρικά αντικείμενα.
    boolean canFire = true; // Έλεγχος εάν ο παίκτης μπορεί να πυροβολήσει ξανά.

    public void act()   // Μέθοδος που εκτελείται κατά την εκτέλεση του προγράμματος.
    {
        jump(); // Κλήση της μεθόδου για την υλοποίηση των άλματων.
        checkFalling(); // Έλεγχος για πτώση του παίκτη.
        fall(); // Κλήση της μεθόδου για την υλοποίηση της πτώσης.
        moveAround(); // Κλήση της μεθόδου για την κίνηση του παίκτη.
        fireBullets(); // Κλήση της μεθόδου για τον πυροβολισμό.
        checkWinCondition(); // Έλεγχος για το αν ο παίκτης έχει φτάσει στο τέλος.
        checkEnemyCollision(); // Έλεγχος για συγκρούσεις με εχθρικά αντικείμενα.
    }
    // Μέθοδος πυροβολισμού. Δημιουργεί ένα νέο αντικείμενο Bullets και το προσθέτει στον κόσμο.
    public void fireBullets()
    {
        MouseInfo mouse = Greenfoot.getMouseInfo();
        if (mouse != null && Greenfoot.mousePressed(null) && canFire == true)
        {
            Bullets bullets = new Bullets();
            getWorld().addObject(bullets, getX(), getY());
            bullets.turnTowards(mouse.getX(), mouse.getY());
            canFire = true;
        }
    }
    // Μέθοδος κίνησης του παίκτη προς τα δεξιά ή αριστερά, ανάλογα με το πλήκτρο που πατά ο χρήστης.
    public void moveAround()
    {
        if (Greenfoot.isKeyDown("right"))
        {
            setLocation(getX() + speed, getY());
        }
        if (Greenfoot.isKeyDown("left"))
        {
            setLocation(getX() - speed, getY());
        }
    }
    // Μέθοδος που εκτελεί την κίνηση προς τα κάτω (πτώση) του παίκτη.
    public void fall()
    {
        setLocation(getX(), getY() + vSpeed);
    }
    // Μέθοδος που ελέγχει αν ο παίκτης είναι σε κατάσταση πτώσης.
    public void checkFalling()
    {
        if (!isTouching(Ground.class))
        {
            vSpeed++;
        }
        else if (isTouching(Ground.class))
        {
            setLocation(getX(), getY() - 1);
            vSpeed = 0;
        }
    }
    // Μέθοδος που υλοποιεί την λειτουργία της άλματος του παίκτη.
    public void jump()
    {
        if (Greenfoot.isKeyDown("up"))
        {
            vSpeed = -10;
        }
    }
    // Μέθοδος που ελέγχει αν ο παίκτης έχει συγκρουστεί με εχθρικά αντικείμενα.
    private void checkEnemyCollision()
    {
        Actor enemy = getOneIntersectingObject(Enemy.class);
        Actor enemy2 = getOneIntersectingObject(Enemy2.class);
        if (((enemy != null) || (enemy2 != null)) && !checkEnemyCollision)
        {
            lives--;
            checkEnemyCollision = true;
        }
        else if (isTouching(Enemy.class) || isTouching(Enemy2.class))
        {
            checkEnemyCollision = false;
        }
        if (lives <= 0)
        {
            getWorld().addObject(new YouLost(), getWorld().getWidth() / 2, getWorld().getHeight() / 2);
            Greenfoot.stop();
            getWorld().removeObject(this);
        }
    }
```

```
    // Μέθοδος που ελέγχει αν ο παίκτης έχει φτάσει στο τέλος του παιχνιδιού.
    private void checkWinCondition()
    {
        if (getX() >= 2999)
        {
            getWorld().addObject(new YouWin(), getWorld().getWidth() / 2, getWorld().getHeight() / 2);
            Greenfoot.stop();
        }
    }
}
```

Η κλάση Bullets αποτελεί μια υποκλάση της κλάσης Actor στο πλαίσιο της βιβλιοθήκης Greenfoot στη γλώσσα προγραμματισμού Java. Στον κώδικα, κάθε αντικείμενο αυτής της κλάσης αναπαριστά ένα βλήμα που πυροβολείται από τον παίκτη. Η ταχύτητα του βλήματος ορίζεται στη μεταβλητή speed. Κατά τη δημιουργία του, το μέγεθος του βλήματος καθορίζεται μέσω της μεθόδου getImage().scale(30, 30) για να είναι ορατό και ευδιάκριτο στην οθόνη. Η κίνηση του βλήματος προς την κατεύθυνση του ποντικιού υλοποιείται με τη χρήση της μεθόδου turnTowards(Greenfoot.getMouseInfo().getX(), Greenfoot.getMouseInfo().getY()), ενώ η μέθοδος move(speed) επιτρέπει την κίνησή του στη δοθείσα ταχύτητα. Αυτή η κλάση είναι σημαντική για την υλοποίηση του πυροβολικού του παίκτη και του τρόπου που αντιδρά στην αλληλεπίδραση με άλλα αντικείμενα στο παιχνίδι.

```
import greenfoot.*;
// Κλάση που αντιπροσωπεύει τα βλήματα που πυροβολούνται από τον παίκτη.
public class Bullets extends Actor
{
    int speed = 10; // Ταχύτητα κίνησης των βλημάτων.

    // Κατασκευαστής της κλάσης. Ορίζει το μέγεθος των βλημάτων κατά τη δημιουργία τους.
    public Bullets()
    {
       getImage().scale(30, 30);
    }
    // Μέθοδος που εκτελείται κατά την εκτέλεση του προγράμματος.
    public void act()
    {
        turnToMouse(); // Κλήση της μεθόδου για τον προσανατολισμό προς το ποντίκι του ποντικιού.
        move(speed); // Κίνηση του βλήματος με βάση την ταχύτητα.
    }
    // Μέθοδος που προσανατολίζει το βλήμα προς τη θέση του ποντικιού.
    public void turnToMouse()
    {
        turnTowards(Greenfoot.getMouseInfo().getX(), Greenfoot.getMouseInfo().getY());
    }
}
```

Οι κλάσεις Enemy και Enemy2 αντιπροσωπεύουν τους εχθρούς στο παιχνίδι, είναι υποκλάσεις της κλάσης Actor στο πλαίσιο της βιβλιοθήκης Greenfoot σε Java. Κάθε αντικείμενο αυτών των κλάσεων υπάρχει μια ταχύτητα (speed), μια μεταβλητή count που χρησιμοποιείται για την προσδιορισμό του χρόνου και μια μεταβλητή health που αντιπροσωπεύει την υγεία του εχθρού. Οι μέθοδος Enemy και Enemy2 δεν είναι ορατές, καθώς το όνομα της πρέπει να είναι ίδιο με το όνομα της κλάσης και εδώ η λέξη "void" δηλώνει τύπο επιστροφής, ενώ θα έπρεπε να είναι "public Enemy()" και είναι "public Enemy2()".

Η μέθοδος act καλείται κατά τη διάρκεια κάθε κύκλου του παιχνιδιού και αυξάνει τον μετρητή count, καλώντας επίσης τις μεθόδους moveAround και hitByBullets. Η moveAround κινεί τον εχθρό προς τα δεξιά για τα πρώτα 60 frames, και μετά αλλάζει την κατεύθυνση και καθρεφτίζει την εικόνα του. Η hitByBullets ελέγχει εάν το αντικείμενο είναι σε επαφή με βλήματα (Bullets) και μειώνει την υγεία του εχθρού αν είναι ταυτόχρονα ενεργή η μεταβλητή hitBullets. Εάν η υγεία του εχθρού πέσει κάτω από το μηδέν, το εχθρικό αντικείμενο αφαιρείται από τον κόσμο του παιχνιδιού.

```
import greenfoot.*;
public class Enemy extends Actor
{
    int speed = 5;            // Η ταχύτητα του εχθρού
    int count = 0;             // Μετρητής χρησιμοποιούμενος για τον προσδιορισμό του χρόνου
    int health = 5;            // Η υγεία του εχθρού
    boolean hitBullets = false; // Flag που δείχνει αν ο εχθρός έχει πληγωθεί από βολή
    public void Enemy()
    {
        getImage().mirrorHorizontally();  // Καθρεφτίζει την εικόνα του εχθρού
    }
    public void act()
    {
        count++;
        moveAround();
        hitByBullets();
    }
    public void moveAround()
    {
        if (count < 60)
        {
            setLocation(getX() + speed,  getY());  // Κίνηση προς τα δεξιά για τα πρώτα 60 frames
        } else
        {
            speed = -speed;                    // Αλλαγή κατεύθυνσης μετά από 60 frames
            getImage().mirrorHorizontally();    // Καθρεφτίζει ξανά την εικόνα
            count = 0;
        }
    }
    public void hitByBullets()
    {
        Actor bullets = getOneIntersectingObject(Bullets.class);
        if ((bullets != null) && !hitBullets)
        {
            health--;               // Μείωση της υγείας του εχθρού αν πληγώσει από βολή
            hitBullets = true;       // Ορισμός της flag ως ενεργή
            getWorld().removeObject(bullets);  // Αφαίρεση του βλήματος από τον κόσμο
        } else if (isTouching(Bullets.class))
        {
            hitBullets = false;      // Αν δεν πληγώθηκε, απενεργοποίηση της flag
        }
        if (health <= 0)
        {
            getWorld().removeObject(this);  // Αφαίρεση του εχθρού από τον κόσμο αν η υγεία πέσει κάτω από το μηδέν
        }
    }
}
```

```java
import greenfoot.*;
public class Enemy2 extends Actor
{
    int speed = 2;              // Η ταχύτητα του εχθρού
    int count = 0;              // Μετρητής χρησιμοποιούμενος για τον προσδιορισμό του χρόνου
    int health = 3;             // Η υγεία του εχθρού
    boolean hitBullets = false;  // Flag που δείχνει αν ο εχθρός έχει πληγωθεί από βολή

    public void Enemy2()
    {
        getImage().mirrorHorizontally();  // Καθρεφτίζει την εικόνα του εχθρού
    }
    public void act()
    {
        count++;
        moveAround();
        hitByBullets();
    }
    public void moveAround()
    {
        if (count < 60) {
            setLocation(getX() + speed,  getY());  // Κίνηση προς τα δεξιά για τα πρώτα 60 frames
        } else {
            speed = -speed;                        // Αλλαγή κατεύθυνσης μετά από 60 frames
            getImage().mirrorHorizontally();    // Καθρεφτίζει ξανά την εικόνα
            count = 0;
        }
    }
    public void hitByBullets()
    {
        Actor bullets = getOneIntersectingObject(Bullets.class);
        if ((bullets != null) && !hitBullets)
        {
            health--;                // Μείωση της υγείας του εχθρού αν πληγώσει από βολή
            hitBullets = true;       // Ορισμός της flag ως ενεργή
            getWorld().removeObject(bullets);  // Αφαίρεση του βλήματος από τον κόσμο
        } else if (isTouching(Bullets.class))
        {
            hitBullets = false;      // Αν δεν πληγώθηκε, απενεργοποίηση της flag
        }
        if (health <= 0)
        {
            getWorld().removeObject(this);  // Αφαίρεση του εχθρού από τον κόσμο αν η υγεία πέσει κάτω από το μηδέν
        }
    }
}
```

Οι κώδικες που παρέχονται παρακάτω αναπαριστούν τις κλάσεις YouWin και YouLost σε ένα παιχνίδι με το πλαίσιο εργασίας Greenfoot. Οι κλάσεις δεν περιέχουν καμία ενέργεια (act() μέθοδος) στην παρούσα μορφή. Αυτό σημαίνει ότι δεν κάνουν τίποτα κατά τη διάρκεια κάθε κύκλου ενημέρωσης του παιχνιδιού.

Οι κλάσεις YouWin και YouLost χρησιμοποιούνται για να εμφανίζουν ένα μήνυμα ή εικόνα που δείχνει ότι ο παίκτης κέρδισε ή έχασε το παιχνίδι, αλλά όπως αναφέρεται, η act() μέθοδος δεν περιέχει κώδικα.

Εάν θέλετε να προσθέσετε λειτουργίες στις κλάσεις YouWin και YouLost, μπορείτε να προσθέσετε κώδικα στη μέθοδο act(). Για παράδειγμα, μπορείτε να προσθέσετε κώδικα για το κλείσιμο του παιχνιδιού όταν ο παίκτης κερδίζει ή χάνει και οποιαδήποτε άλλη λειτουργία που επιθυμείτε.

```java
import greenfoot.*;
public class YouWin extends Actor
{

    public void act()
    {
        // Add your action code here.
    }
}
```

```
import greenfoot.*;
public class YouLost extends Actor
{
    public void act()
    {
        // Add your action code here.
    }
}
```

Στη γλώσσα προγραμματισμού Java, οι λέξεις-κλειδιά private και public χρησιμοποιούνται για τον έλεγχο της πρόσβασης σε μέλη (μεταβλητές, μέθοδοι, κ.λπ.) ενός αντικειμένου ή μιας κλάσης. Παρακάτω παρουσιάζεται η διαφορά μεταξύ τους:

Private:

Όταν μια μεταβλητή ή μια μέθοδος δηλώνεται ως private, αυτό σημαίνει ότι είναι προσβάσιμη μόνο εντός της ίδιας κλάσης.

Δεν μπορεί να προσπελαστεί από άλλες κλάσεις ή αντικείμενα που δημιουργούνται από άλλες κλάσεις.

Παρέχει υψηλότερο επίπεδο απομόνωσης και ελέγχου των δεδομένων, καθώς περιορίζει την πρόσβαση σε εσωτερικά μέλη της κλάσης.

Παράδειγμα:

```
public class MyClass
 {
   private int private Variable;
   private void private Method()
{
      // Κώδικας μεθόδου
   }
}
```

Public:

Όταν μια μεταβλητή ή μια μέθοδος δηλώνεται ως public, αυτό σημαίνει ότι είναι προσβάσιμη από οποιοδήποτε άλλο σημείο του προγράμματος.

Παρέχει ευρεία πρόσβαση στα μέλη της κλάσης και μπορεί να χρησιμοποιηθεί από άλλες κλάσεις ή αντικείμενα.

Είναι συχνά χρήσιμη για μέλη που πρέπει να είναι προσβάσιμα από άλλα τμήματα του προγράμματος.

Παράδειγμα:

```
public class MyClass
{
   public int public Variable;
   public void public Method()
 {
      // Κώδικας μεθόδου
   }
}
```

Η επιλογή μεταξύ private και public εξαρτάται από τη σχεδίαση του προγράμματος και το επίπεδο ελέγχου που επιθυμεί ο προγραμματιστής να διατηρήσει στον κώδικά του.

Αναφορές

- https://www.youtube.com/@CodingClub
- https://www.youtube.com/watch?v=LT6IPr2W8dw (ανακτήθηκε στις 14/11/2023)