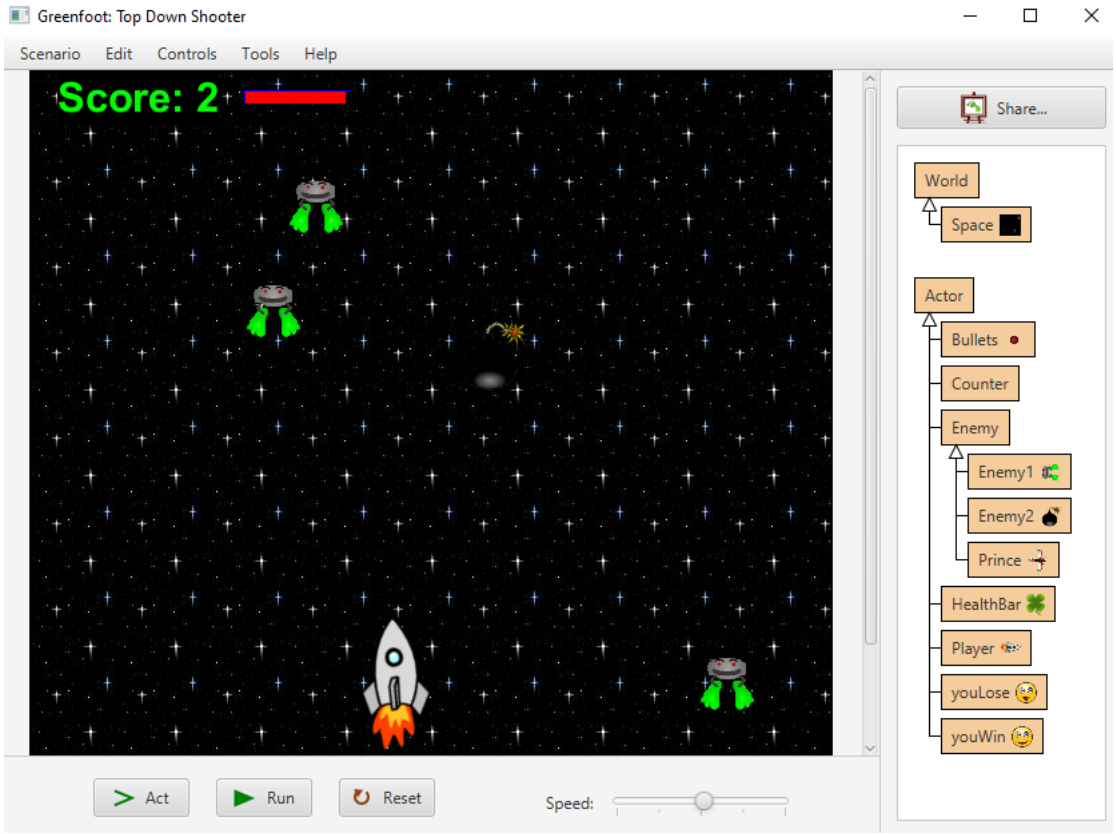
Άσκηση 7η

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



Ο παρακάτω κώδικας αναπαριστά έναν κόσμο (world) στο παιχνίδι που φτιάχνεται με τη βιβλιοθήκη Greenfoot σε Java.

Αυτός ο κώδικας αντιπροσωπεύει ένα απλό παιχνίδι όπου ο παίκτης κινείται και προσπαθεί να αποφύγει τους εχθρούς. Ο κόσμος δημιουργείται με μία μπάρα υγείας για τον παίκτη, ένα μετρητή σκορ και εμφανίζει εχθρούς τύπου 1 και 2. Επίσης, εμφανίζει τον πρίγκιπα σε διάφορα επίπεδα, ανάλογα με το σκορ του παίκτη.

```
import greenfoot.*;

/**
 * Write a description of class Space here.
 *
 * Kaffetzis Vasileios
 * 02/11/2023
 */
public class Space extends World
{
    HealthBar Healthbar = new HealthBar (); // Δημιουργία αντικειμένου HealthBar για την υγεία του παίκτη.
    private Counter counter = new Counter(); // Δημιουργία αντικειμένου Counter για τη μέτρηση του σκορ.
    boolean princeLevel = false;
    boolean princeLeve2 = false; // Μεταβλητές για τον έλεγχο του εμφανιζόμενου επιπέδου του πρίγκιπα.
    boolean princeLeve3 = false;
    public Space() // Κατασκευαστής της κλάσης Space.
    {
        super(600, 600, 1);  // Δημιουργία νέου κόσμου με διαστάσεις 600x600 κελιά και μεγέθος κελιού 1x1 pixel.
        prepare(); // Κλήση της μεθόδου prepare() για την προετοιμασία του κόσμου.
    }
    public Counter getCounter()
    {
        return counter; // Μέθοδος που επιστρέφει το αντικείμενο Counter.
    }
    public void increaseScore() // Μέθοδος που αυξάνει το σκορ του παίκτη.
    {
        counter.addScore();
    }
    public HealthBar getHealthBar() // Μέθοδος που επιστρέφει το αντικείμενο HealthBar.
    {
        return Healthbar;
    }
    public void act()   // Μέθοδος που καλείται κάθε φορά που εκτελείται ένα βήμα στον κόσμο.
    {
        addEnemy1();    // Προσθήκη εχθρού τύπου 1.
        addEnemy2();    // Προσθήκη εχθρού τύπου 2.
        Prince();       // Ελέγχει την εμφάνιση του πρίγκιπα.
    }
    public void Prince()     // Μέθοδος που ελέγχει την εμφάνιση του πρίγκιπα ανάλογα με το σκορ του παίκτη.
    {
        if (counter.score == 15 || counter.score == 16)
        {
        if (princeLevel == false) // Επίπεδο 1
        {
            addObject(new Prince(), 300, 0);
            princeLevel = true;
        }
        }
        if (counter.score == 30 || counter.score == 31)
        {
        if (princeLeve2 == false) // Επίπεδο 2
        {
            addObject(new Prince(), 300, 0);
            princeLeve2 = true;
        }
        }
        if (counter.score == 48 || counter.score == 49)
        {
        if (princeLeve3 == false) // Επίπεδο 3
        {
            addObject(new Prince(), 300, 0);
            princeLeve3 = true;
        }
        }
    }
    public void addEnemy1() // Μέθοδος που προσθέτει έναν εχθρό τύπου 1 στον κόσμο.
    {
      if (Greenfoot.getRandomNumber (150) < 1) // Συχνότητα εμφάνισης στον κόσμο
        {
            addObject (new Enemy1(), Greenfoot.getRandomNumber(600), 0);
        }
    }
    public void addEnemy2() // Μέθοδος που προσθέτει έναν εχθρό τύπου 2 στον κόσμο.
    {
        if (Greenfoot.getRandomNumber (180) < 1) // Συχνότητα εμφάνισης στον κόσμο
        {
            addObject (new Enemy2(), Greenfoot.getRandomNumber(600), 0);
        }
    }
    private void prepare() // Προετημασία για εισαγωγή αντικειμένων στον κόσμο
    {
        addObject(counter, 80, 20);
        addObject (Healthbar, 200, 20);
        Player player = new Player();
        addObject(player,252,510);
        player.setLocation(270,490);
    }
}
```

Ο παρακάτω κώδικας αντιπροσωπεύει την κλάση Player σε ένα παιχνίδι που χρησιμοποιεί τη βιβλιοθήκη Greenfoot σε Java.

Αυτή η κλάση αντιπροσωπεύει τον παίκτη στο παιχνίδι. Ο παίκτης μπορεί να κινείται προς τα αριστερά , τα δεξιά, τα πάνω και τα κάτω. Επίσης, ο παίκτης μπορεί να πυροβολήσει πυραύλους όταν πατάει το πλήκτρο "space". Η κλάση ελέγχει επίσης τη σύγκρουση με εχθρικά αντικείμενα και μειώνει την μπάρα υγείας του παίκτη αν υπάρχει σύγκρουση. **Η μεταβλητή delay χρησιμοποιείται για να ελέγχει το χρονικό διάστημα καθυστέρησης μεταξύ των επαναλήψεων των ενεργειών του παίκτη**.

Σε αυτό το σενάριο, ο παίκτης και οι εχθρικοί χαρακτήρες (Enemy) ανήκουν στην ίδια ιεραρχία κληρονομίας (Actor), και έχει σχεδιαστεί η λογική για τις αλληλεπιδράσεις μεταξύ τους. Οι ενέργειες του παίκτη, όπως η κίνηση και ο πυροβολισμός, ελέγχονται από τα πλήκτρα του πληκτρολογίου και εκτελούνται σε κάθε βήμα του παιχνιδιού (act method).

```
import greenfoot.*;
public class Player extends Actor
{
    boolean canFire = true; // Μεταβλητή που υποδεικνύει εάν ο παίκτης μπορεί να πυροβολήσει.
    int space1 = 0;         // Μεταβλητή που χρησιμοποιείται για τον έλεγχο της καθυστέρησης των πυροβολισμών.
    private int delay = 30; // Το χρονικό διάστημα καθυστέρησης σε κύκλους ενέργειας
    public Player ()        // Κατασκευαστής της κλάσης Player, ορίζει την αρχική περιστροφή του παίκτη.
    {
        setRotation(270);
    }
    public void act()       // Μέθοδος που εκτελείται κατά την κάθε ενέργεια του παίκτη.
    {
        moveAround();       // Μέθοδος για την κίνηση του παίκτη.
        Bullets();          // Μέθοδος για την δημιουργία πυραύλων
        checkWinCondition();    // Έλεγχος εάν ο παίκτης έχει κερδίσει το παιχνίδι.
        checkCollision();    // Έλεγχος σύγκρουσης με εχθρικά αντικείμενα.

    }
    public void moveAround()    // Μέθοδος για την κίνηση του παίκτη προς τα αριστερά,τα δεξιά,τα πάνω και τα κάτω.
    {
        if (Greenfoot.isKeyDown ("right"))
        {
            setLocation(getX()+5,getY());
        }
        if (Greenfoot.isKeyDown ("left"))
        {
            setLocation(getX()-5,getY());
        }
        if (Greenfoot.isKeyDown ("down"))
        {
            setLocation(getX(),getY()+5);
        }
        if (Greenfoot.isKeyDown ("up"))
        {
            setLocation(getX(),getY()-5);
        }

    }
    public void Bullets()       // Μέθοδος για τη δημιουργία πυραύλων όταν πατιέται το πλήκτρο "space".
    {
        if (Greenfoot.isKeyDown("space") && canFire == true)
        {
            getWorld().addObject(new Bullets(), getX(), getY()-40);
            canFire = false;
        }
        else if (!Greenfoot.isKeyDown("space"))
        {
            canFire = true;
        }
    }
    private void checkWinCondition()     // Μέθοδος για τον έλεγχο αν ο παίκτης έχει κερδίσει το παιχνίδι (σκοτώσει 50 εχθρούς).
    {
        Space world = (Space)getWorld(); // Αναφορά στο αντικείμενο Space
        Counter counter = world.getCounter(); // Λαμβάνουμε τον μετρητή από τον κόσμο
        counter.youWin();               // Έλεγχος αν έχει σκοτώσει 50 εχθρικά αντικείμενα και ο παίκτης έχει κερδίσει
    }
    private void checkCollision()       // Μέθοδος για τον έλεγχο σύγκρουσης με εχθρικά αντικείμενα.
    {
        Actor enemy = getOneIntersectingObject(Enemy.class); // Υποθέτοντας ότι η κλάση του εχθρού είναι Enemy
        if (enemy != null && delay <= 0)
        {
        Space world = (Space)getWorld();// Αναφορά στο αντικείμενο Space
        HealthBar healthBar = world.getHealthBar();// Λήψη της μπάρας ζωής από τον κόσμο
        healthBar.loseHealth(); // Μείωση της μπάρας ζωής κατά ένα
        delay = 30; // Επαναφορά της καθυστέρησης
        }
        else if (delay > 0)
        {
        delay--; // // Μείωση της καθυστέρησης κάθε κύκλο
        }
    }

}
```

Αυτός ο κώδικας αντιπροσωπεύει την κλάση Enemy σε ένα παιχνίδι που χρησιμοποιεί τη βιβλιοθήκη Greenfoot σε Java.

Σε αυτή την κλάση, ο εχθρός (Enemy) μπορεί να κινηθεί προς τα κάτω κάθε φορά που κληθεί η μέθοδος **moveEnemy()**. **Αν η θέση Y του εχθρού είναι μεγαλύτερη ή ίση του 599, ο εχθρός αφαιρείται από τον κόσμο μέσω της μεθόδου removeEnemy()**. Οι λεπτομέρειες της λογικής του παιχνιδιού που χρησιμοποιεί αυτήν την κλάση δεν είναι πλήρως προσδιορισμένες σε αυτόν τον κώδικα, αλλά αυτά τα δύο μέρη του κώδικα περιγράφουν την κίνηση και την αφαίρεση του εχθρού αντίστοιχα.

```
import greenfoot.*;
public class Enemy extends Actor
{
    public void act()       // Μέθοδος που εκτελείται κατά την κάθε ενέργεια του εχθρού.
    {
        //
    }
    public void moveEnemy()  // Μέθοδος για την κίνηση του εχθρού προς τα κάτω.
    {
       setLocation(getX(),getY()+3);
    }
    public void removeEnemy() // Μέθοδος για την αφαίρεση του εχθρού από τον κόσμο όταν φτάσει σε συγκεκριμένη θέση Y.
    {
        if (getY() >= 599)
        {
            getWorld().removeObject(this);
        }
    }
}
```

Στην κλάση **Enemy1**, που είναι υποκλάση της κλάσης **Enemy**, υλοποιούνται λειτουργίες για έναν εχθρό σε ένα παιχνίδι στο πλαίσιο του περιβάλλοντος προγραμματισμού Greenfoot. **Η κλάση περιλαμβάνει μεταβλητές όπως η isHit, που δείχνει εάν το Enemy1 έχει πληγεί**, και **η timesHit που κρατάει τον αριθμό των φορών που έχει πληγεί**. Υλοποιεί μεθόδους όπως η **act()**, που καθορίζει τη συμπεριφορά του εχθρού κατά τη διάρκεια της εκτέλεσης του προγράμματος. Επίσης, περιλαμβάνει **μεθόδους όπως η hitByBullets(), που ελέγχει αν το Enemy1 πληγώθηκε από σφαίρες και αυξάνει το σκορ του παίκτη**, και η **checkBoundary()**, **που ελέγχει τα όρια του περιβάλλοντος του παιχνιδιού και μειώνει την μπάρα υγείας του παίκτη όταν το Enemy1 πληγώνεται ή φτάνει στα όρια**. Επιπλέον, **υπάρχει μια μη χρησιμοποιημένη μέθοδος checkPlayerCollision() που ελέγχει σύγκρουση με τον παίκτη** αλλά δεν καλείται από κάπου στον κώδικα. Εάν αυτή η μέθοδος χρειάζεται να καλείται από κάπου αλλού στο πρόγραμμά. Στο σύνολο, αυτή η κλάση υλοποιεί τη λογική της συμπεριφοράς ενός εχθρικού χαρακτήρα στο παιχνίδι.

```java
import greenfoot.*;

public class Enemy1 extends Enemy
{
    private boolean isHit = false;  // Μεταβλητή που δηλώνει εάν το Enemy1 έχει πληγεί
    int timesHit = 1;               // Μεταβλητή που κρατάει τον αριθμό των φορών που έχει πληγεί το Enemy1

    public Enemy1()
    {
        setRotation(90);            // Ορίζει την αρχική περιστροφή του Enemy1 στα 90 μοίρες (κάθετη κίνηση)
    }
      public void act()
    {
        moveEnemy();                // Καλεί τη μέθοδο για την κίνηση του Enemy1
        hitByBullets();             // Καλεί τη μέθοδο για τον έλεγχο σύγκρουσης με σφαίρες
        checkBoundary();            // Καλεί τη μέθοδο για τον έλεγχο σύγκρουσης με τα όρια του παιχνιδιού

    }
    public void hitByBullets()
    {
        Actor Bullets = getOneIntersectingObject(Bullets.class);        // Ελέγχει αν υπάρχει σύγκρουση με αντικείμενο τύπου Bullets
        if (Bullets != null)
        {
            getWorld().removeObject(Bullets);     // Αφαιρεί το αντικείμενο Bullets από τον κόσμο
            isHit = true;                        // Το Enemy1 έχει πληγεί
            World world = getWorld();
            Space space1 = new Space();
            Counter counter = space1.getCounter();
            counter.addScore();                   // Αυξάνει το σκορ του παίκτη όταν το Enemy1 πληγώνεται
            ((Space) getWorld()).increaseScore();   // Αυξάνει το σκορ του παίκτη (πιθανόν να υπάρχει μέθοδος increaseScore στην κλάση Spa
        }
    }
        public void checkBoundary()

    {
        if (isHit || getY()>= 599)
        {
            World world = getWorld();
            Space space1 = new Space();
            HealthBar healthbar = space1.getHealthBar();
            healthbar.loseHealth();                 // Μειώνει την μπάρα υγείας όταν το Enemy1 πληγώνεται ή φτάνει στο κάτω όριο του παιχνι
            getWorld().removeObject(this);              // Αφαιρεί το Enemy1 από τον κόσμο
        }
        else if (isHit || getY()== 0)
        {
            getWorld().removeObject(this); // Αφαιρεί το Enemy1 από τον κόσμο όταν φτάσει στο πάνω όριο του παιχνιδιού
        }
    }
    public void checkPlayerCollision()
     {
        Player player = (Player) getOneIntersectingObject(Player.class);
        if (player != null)
        {
            getWorld().removeObject(this); // Αν εντοπιστεί σύγκρουση με τον Player, αφαιρέστε το Enemy1
        }
    }
}
```

Ο κώδικας της κλάσης **Enemy2** αντιπροσωπεύει έναν εχθρικό χαρακτήρα σε ένα παιχνίδι που έχει υλοποιηθεί με το περιβάλλον προγραμματισμού Greenfoot. Το αντικείμενο **Enemy2** μπορεί να κινηθεί στην οθόνη, να ανιχνεύει συγκρούσεις με σφαίρες (αντικείμενα τύπου **Bullets**) και να μειώνει την μπάρα υγείας του παίκτη όταν πληγώνεται. Το **Enemy2** έχει έναν αρχικά ορισμένο αριθμό φορών που πρέπει να πληγεί προτού εξαφανιστεί, και αυξάνει το σκορ του παίκτη κάθε φορά που πληγώνεται από μια σφαίρα. Επιπλέον, ελέγχει αν υπάρχει σύγκρουση με τον παίκτη και εξαφανίζεται αν αυτή η σύγκρουση συμβεί. Ο κώδικας είναι σχεδιασμένος για να υλοποιεί τη λογική ενός εχθρικού χαρακτήρα και την αλληλεπίδρασή του με τα υπόλοιπα αντικείμενα στο παιχνίδι.

```java
import greenfoot.*;

public class Enemy2 extends Enemy
{
    private boolean isHit = false;  // Μεταβλητή που δηλώνει εάν το Enemy2 έχει πληγεί
    int timesHit = 2;               // Αρχικός αριθμός φορών που το Enemy2 πρέπει να πληγεί προτού εξαφανιστεί

    public void act()
    {
        moveEnemy();        // Καλεί τη μέθοδο για την κίνηση του Enemy2
        hitByBullets();     // Καλεί τη μέθοδο για τον έλεγχο σύγκρουσης με σφαίρες
        checkBoundary();    // Καλεί τη μέθοδο για τον έλεγχο των ορίων του παιχνιδιού
    }
    public void hitByBullets ()
    {
        Actor Bullets = getOneIntersectingObject(Bullets.class); // Ελέγχει αν υπάρχει σύγκρουση με αντικείμενο τύπου Bullets
        if (Bullets != null)
        {
            getWorld().removeObject(Bullets);   // Αφαιρεί το αντικείμενο Bullets από τον κόσμο
            timesHit--;                         // Μειώνει τις φορές που το Enemy2 πρέπει να πληγεί
            World world = getWorld();
            Space space1 = new Space();
            Counter counter = space1.getCounter();
            counter.addScore();                 // Αυξάνει το σκορ του παίκτη
            ((Space) getWorld()).increaseScore(); // Αυξάνει το σκορ του παίκτη (πιθανόν να υπάρχει μέθοδος increaseScore στην κλάση Space
        }
        if (timesHit <= 0)
        {
            getWorld().removeObject(this);          // Αφαιρεί το Enemy2 από τον κόσμο όταν πληγώθηκε ορισμένος αριθμός φορών
        }
    }
    public void checkBoundary()
    {
        if (getWorld() != null && (isHit || getY() >= 599))
                // Ελέγχει πρώτα αν το αντικείμενο Enemy2 υπάρχει ακόμη στον κόσμο (getWorld() != null)
                // Μειώνει τη μπάρα υγείας του Player, πολλές φορές όταν το Enemy2 τον ακουμπάει ή φτάνει στο κάτω όριο του παιχνιδιού
        {
            World world = getWorld();
            Space space1 = new Space();
            HealthBar healthbar = space1.getHealthBar();
            healthbar.loseHealth();
            healthbar.loseHealth(); // Μειώνει τη μπάρα υγείας του παίκτη όταν το Enemy2 πληγώνεται ή φτάνει στο κάτω όριο του παιχνιδιού
            getWorld().removeObject(this); // Αφαιρεί το Enemy2 από τον κόσμο
        }
    }
    public void checkPlayerCollision()
    {
        Player player = (Player) getOneIntersectingObject(Player.class); // Ελέγχει αν υπάρχει σύγκρουση με τον παίκτη
        if (player != null)
        {
            getWorld().removeObject(this);              // Αφαιρεί το Enemy2 από τον κόσμο όταν συγκρουστεί με τον παίκτη
        }
    }
}
```

Σε αυτόν τον κώδικα, η κλάση **Prince** αντιπροσωπεύει έναν εχθρικό χαρακτήρα στο παιχνίδι. Ο **Prince** μπορεί να μετακινηθεί στην οθόνη, να ανιχνεύει συγκρούσεις με σφαίρες, να μειώνει τη μπάρα υγείας του παίκτη όταν πληγώνεται, και να εξαφανίζεται όταν πληγώνεται επαρκής αριθμός φορών ή φτάσει στα κάτω όρια του παιχνιδιού. Η κλάση παρέχει λειτουργικότητα για τη διαχείριση των αλληλεπιδράσεων με τον παίκτη και τους εχθρικούς χαρακτήρες στο παιχνίδι.

```java
import greenfoot.*;

public class Prince extends Enemy
{
    int timesHit = 10;     // Αρχικός αριθμός φορών που το Prince πρέπει να πληγεί προτού εξαφανιστεί
    boolean isHit = false; // Μεταβλητή που δηλώνει εάν το Prince έχει πληγεί

    public Prince ()
    {
        setRotation(90);  // Ορίζει την αρχική περιστροφή του Prince στα 90 μοίρες (κάθετη κίνηση)
    }
    public void act()
    {
        moveEnemy();     // Καλεί τη μέθοδο για την κίνηση του Prince
        hitByBullets();  // Καλεί τη μέθοδο για τον έλεγχο σύγκρουσης με σφαίρες
        checkBoundary(); // Καλεί τη μέθοδο για τον έλεγχο των ορίων του παιχνιδιού
    }
```

```
    public void hitByBullets()
    {
    Actor Bullets = getOneIntersectingObject(Bullets.class); // Ελέγχει αν υπάρχει σύγκρουση με αντικείμενο τύπου Bullets
    if (Bullets != null)
        {
        getWorld().removeObject(Bullets);   // Αφαιρεί το αντικείμενο Bullets από τον κόσμο
        timesHit--;   // Μειώνει τις φορές που ο πρίγκιπας πρέπει να πληγεί
        //World world = getWorld();
        //Space space1 = new Space();
        //Counter counter = space1.getCounter();
        //counter.addScore();                 // Αυξάνει το σκορ του παίκτη
        //((Space) getWorld()).increaseScore(); // Αυξάνει το σκορ του παίκτη (πιθανόν να υπάρχει μέθοδος increaseScore στην κλάση Space
        }
    if (timesHit <= 0)
        {
            getWorld().removeObject(this);  // Αν οι φορές που πρέπει να πληγεί είναι μηδέν ή αρνητικές, αφαιρεί τον πρίγκιπα από τον κόσμο
        }
    }
```

Αν βάλω αυτό το πακέτο εντολών, που είναι σε σχόλια, θα ενεργοποιηθεί η δυνατότητα της αύξησης του Σκορ για κάθε επαφή της σφαίρας με το Prince. Αυτό θα κάνει το παιχνίδι πιο εύκολο και Δεν το θέλουμε

```
     public void checkBoundary()
     {
         if (getWorld() != null && (isHit || getY() >= 599))
                    // Ελέγχει πρώτα αν το αντικείμενο Prince υπάρχει ακόμη στον κόσμο (getWorld() != null)
                    // Μειώνει τη μπάρα υγείας του Player πολλές φορές, όταν το Prince τον πληγώνει ή φτάνει στο κάτω όριο του παιχνιδιού

         {
             World world = getWorld();
             Space space1 = new Space();
             HealthBar healthbar = space1.getHealthBar();

             healthbar.loseHealth();
             healthbar.loseHealth();
             healthbar.loseHealth();
             healthbar.loseHealth();
             healthbar.loseHealth();     // Για τις 10 επαναλήψεις
             healthbar.loseHealth();     // Μπορούμε να χρησιμοποιήσουμε και Δομή Επανάληψης for....
             healthbar.loseHealth();     // for (int i = 0; i < 10; i++) {healthbar.loseHealth();}
             healthbar.loseHealth();
             healthbar.loseHealth();
             healthbar.loseHealth();
             getWorld().removeObject(this);      // Αφαιρεί το Prince από τον κόσμο
         }
     }
     public void checkPlayerCollision()      // Ελέγχει αν υπάρχει σύγκρουση με τον παίκτη
     {
         Player player = (Player) getOneIntersectingObject(Player.class);
         if (player != null)                 // Αν υπάρξει σύγκρουση
         {
             getWorld().removeObject(this); // Αφαιρεί το Prince από τον κόσμο όταν συγκρουστεί με τον παίκτη
         }
     }

}
```

Αυτή η κλάση **Bullets** υλοποιεί τη λογική για τη συμπεριφορά των σφαιρών στο παιχνίδι. Η **act()** μέθοδος καλεί τις **BulletsMove()** και **removeFromWorld()**. Η **BulletsMove()** μέθοδος κινεί το αντικείμενο σφαίρας πάνω (προς τα πάνω) κατά 5 μονάδες σε κάθε κύκλο εκτέλεσης. Η **removeFromWorld()** μέθοδος ελέγχει αν υπάρχει σύγκρουση με άλλη σφαίρα. Εάν υπάρχει σύγκρουση, αφαιρεί τις δύο σφαίρες από τον κόσμο. Εάν η σφαίρα φτάσει στην κορυφή του παιχνιδιού (όπου Y=0), αφαιρείται επίσης από τον κόσμο. Η κλάση αυτή χρησιμοποιείται για να δημιουργήσει τον χειριστή των σφαιρών στο παιχνίδι, καθώς και να διαχειριστεί την αφαίρεσή τους από τον κόσμο όταν συγκρουστούν ή φτάσουν στα όρια του περιβάλλοντος.

```
import greenfoot.*;

public class Bullets extends Actor
{
    public void act()
    {
        BulletsMove();      // Καλεί τη μέθοδο για την κίνηση της σφαίρας
        removeFromWorld();  // Καλεί τη μέθοδο για τον έλεγχο και την αφαίρεση της σφαίρας από τον κόσμο
    }
    public void BulletsMove()
    {
        setLocation(getX(), getY() - 5);  // Μετακινεί τη σφαίρα πάνω κατά 5 μονάδες σε κάθε κύκλο εκτέλεσης
    }
```

```
    public void removeFromWorld()
    {
        Actor Bullets = getOneIntersectingObject(Bullets.class);  // Ελέγχει αν υπάρχει σύγκρουση με άλλη σφαίρα
        if (Bullets != null)
        {
            getWorld().removeObject(Bullets);  // Αφαιρεί την άλλη σφαίρα από τον κόσμο
            getWorld().removeObject(this);     // Αφαιρεί την τρέχουσα σφαίρα από τον κόσμο
        }
        else if (getY() == 0)
        {
            getWorld().removeObject(this);  // Αφαιρεί την σφαίρα από τον κόσμο όταν φτάσει στην κορυφή του παιχνιδιού (Y=0)
        }
    }
}
```

Η κλάση **Counter** χρησιμοποιείται για να αντιπροσωπεύει το σκορ του παίκτη στο παιχνίδι. Αυτή ενημερώνεται κατάλληλα κάθε φορά που ο παίκτης κερδίζει πόντους (με τη μέθοδο **addScore()**) και ελέγχει αν το σκορ έχει φτάσει τους 50 πόντους, προκειμένου να εμφανίσει το μήνυμα νίκης (**youWin()**) και να σταματήσει το παιχνίδι.

```
import greenfoot.*;
public class Counter extends Actor
{
    int score = 0;                          // Μεταβλητή για την αποθήκευση του σκορ του παίκτη
    public Counter()
    {
        setImage(new GreenfootImage("Score: " + score, 40, Color.GREEN, Color.BLACK)); // Ορίζει την εικόνα του Counter με το αρχικό σκορ
    }
    public void act()
    {
        setImage(new GreenfootImage("Score: " + score, 40, Color.GREEN, Color.BLACK)); // Ενημερώνει την εικόνα του Counter με το τρέχον σ
    }
    public void addScore()
    {
        score++;                           // Αυξάνει το σκορ κατά 1 κάθε φορά που καλείται αυτή η μέθοδος
    }
    public void youWin()
    {
        if (score >= 50)
        {
            getWorld().addObject(new youWin(), 300, 300); // Προσθέτει ένα νέο αντικείμενο youWin() στον κόσμο στις συγκεκριμένες συντεταγ
            Greenfoot.stop();                             // Σταματά το παιχνίδι αν το σκορ είναι μεγαλύτερο ή ίσο με 50
        }
    }
}
```

Η κλάση **HealthBar** χρησιμοποιείται για να αναπαραστήσει την υγεία του παίκτη στο παιχνίδι. Ενημερώνει διαρκώς την εικόνα της μπάρας υγείας (**update()**) και ελέγχει εάν ο παίκτης έχασε (**youLose()**) αν η υγεία φτάσει στο 0. Ο παίκτης χάνει υγεία όταν συγκρούεται με εχθρικούς χαρακτήρες.

```
import greenfoot.*;

public class HealthBar extends Actor
{
    int health = 15;                                    // Αρχική τιμή υγείας
    int healthBarWidth = 80;                            // Πλάτος της μπάρας υγείας
    int healthBarHeight = 10;                           // Ύψος της μπάρας υγείας
    int pixelsPerHealthPoint = healthBarWidth / health; // Υπολογισμός πίξελ ανά μονάδα υγείας

    public HealthBar()
    {
        update();       // Καλεί τη μέθοδο update για την αρχική ενημέρωση της μπάρας υγείας
    }
    public void act()
    {
        update();       // Καλεί τη μέθοδο update κάθε κύκλο εκτέλεσης για να ενημερώσει την εικόνα της μπάρας υγείας
        youLose();      // Καλεί τη μέθοδο youLose για να ελέγξει αν ο παίκτης έχασε
    }
```

```
    public void update()
    {
        setImage(new GreenfootImage(healthBarWidth + 2, healthBarHeight + 2)); // Δημιουργεί μια νέα εικόνα για την μπάρα υγείας
        GreenfootImage myImage = getImage();
        myImage.setColor(Color.BLUE);
        myImage.drawRect(0, 0, healthBarWidth + 1, healthBarHeight + 1);    // Σχεδιάζει το περίγραμμα της μπάρας υγείας
        myImage.setColor(Color.RED);
        myImage.fillRect(1, 1, health * pixelsPerHealthPoint, healthBarHeight); // Σχεδιάζει το γεμάτο μέρος της μπάρας υγείας, βάσει της
    }
    public void loseHealth()
    {
        health--; // Μειώνει την υγεία κατά 1 μονάδα
    }
    public void youLose()
    {
        if (health == 0)
        {
            getWorld().addObject(new youLose(), 300, 300); // Προσθέτει ένα νέο αντικείμενο youLose() στον κόσμο στις συγκεκριμένες συντετ
            Greenfoot.stop();                               // Σταματά το παιχνίδι αν η υγεία φτάσει στο 0
        }
    }
}
```

Η κλάση **youWin** είναι υπεύθυνη για την εμφάνιση του μηνύματος "You Win!" όταν ο παίκτης κερδίσει το παιχνίδι. Σε αυτήν την περίπτωση, δεν έχει ορισμένη συγκεκριμένη συμπεριφορά κατά τη διάρκεια του παιχνιδιού, άρα η μέθοδος **act()** είναι κενή.

```
import greenfoot.*;

public class youWin extends Actor
{

    public void act()
    {
        // Δεν υπάρχει κώδικας ενέργειας εδώ, καθώς η κλάση αυτή δεν έχει συγκεκριμένη συμπεριφορά κατά τη διάρκεια του παιχνιδιού

    }
}
```

Η κλάση **youLose** υπεύθυνη για την εμφάνιση του μηνύματος "You Lose!" όταν ο παίκτης χάνει το παιχνίδι. Σε αυτήν την περίπτωση, δεν έχει ορισμένη συγκεκριμένη συμπεριφορά κατά τη διάρκεια του παιχνιδιού, άρα η μέθοδος **act()** είναι κενή.

```
import greenfoot.*;

public class youLose extends Actor
{

    public void act()
    {
        // Δεν υπάρχει κώδικας ενέργειας εδώ, καθώς η κλάση αυτή δεν έχει συγκεκριμένη συμπεριφορά κατά τη διάρκεια του παιχνιδιού

    }
}
```

Αναφορές

- https://www.youtube.com/@CodingClub
- https://www.youtube.com/watch?v=LT6IPr2W8dw (ανακτήθηκε στις 02/11/2023)