Platform Jumper: Running, Falling, Jumping

With thanks to Michael Kolling
Pengu!

We will be implementing a small segment of a platform game. In this game, the player controls a character that has to move from one area on the screen to another, while overcoming obstacles such as a gap in the ground.
In this scenario, there are two pieces of ground on either side of the screen, and the penguin can get across by jumping onto a moving cloud. We will implement the running, jumping and falling behavior.
Creating Movement

Click Run. Notice that Pengu can move left/right but does not fall when he leaves ground.
Let’s Make Some Changes

Right Click the Mover Class
Click On Open Editor. Repeat for Pengu class
When Run is Clicked, there is only left/ right movement without Falling or jumping ability.

This is because no method exists in the source code of the Mover class for falling or jumping.

This method would need to be called in the Pengu class.
import greenfoot.*; // (World, Actor, GreenfootImage, and Greenfoot)

/**
 * The class Mover provides some basic movement methods. Use this as a superclass
 * for other actors that should be able to move left and right, jump up and fall
 * down.
 */

class Mover extends Actor
{
    private static final int speed = 7; // running speed (sideways)

    private int vSpeed = 5; // vertical speed

    public void moveRight()
    {
        setLocation( getX(), getY() );
    }
}
public void moveLeft()
{
    setLocation ( getX(), getY() );
}

public void fall()
{
    setLocation ( getX(), getY() + vSpeed);
}
Almost Ready To Go...

For testing purposes, open the Pengu class.

Add the method call `fall();`;

Click Compile and Run.

Notice that Pengu falls right away and the fall looks strange. This is because it is falling at a constant rate instead of accelerating as in a natural fall.
Watch Pengu fall!

Pengu falls off the cliff at a constant rate!

But right now, he falls even if he’s on ground
Creating acceleration during fall

public class Mover extends Actor
{
    private static final int speed = 7; // running speed (sideways)
    private int vSpeed = 0; // current vertical speed
    private static final int acceleration = 2; // down (gravity)

    public void moveRight()
    {
        setLocation ( getX() + speed, getY() );
    }

    public void moveLeft()
    {
        setLocation ( getX() - speed, getY() );
    }

    public void setVSpeed(int speed)
    {
        vSpeed = speed;
    }

    public void fall()
    {
        setLocation ( getX(), getY() + vSpeed);

        vSpeed = vSpeed + acceleration;
    }
}

For a more realistic fall, we need acceleration to simulate gravity (vSpeed needs to increase during the fall)

Open the Mover class

Initialize the variable vSpeed to 0 and create method setVSpeed

Create a new variable for acceleration

Add the following code to the fall() method

Hit Compile button and run.
Preventing falling when on ground

```java
public class Mover extends Actor {
    private static final int acceleration = 2; // down (gravity)
    private static final int speed = 7; // running speed (sideways)

    private int vSpeed = 0; // current vertical speed

    public void moveRight()
    {
        setLocation ( getX() + speed, getY() ) ;
    }

    public void moveLeft()
    {
        setLocation ( getX() - speed, getY() );
    }

    public boolean onGround()
    {
        Object under = getOneObjectAtOffset(0, getImage().getHeight()/2-8, null);
        return under != null;
    }
}
```

Pengu currently falls even if he is standing on ground

We need to check if he’s on ground, (check to see if there is an object immediately under our object of type Pengu) and only fall if he’s not

We can do this using a boolean method called onGround. A boolean tests to see if a condition holds true or not.

Open the Mover class, create the method and implement it as follows

Hit Compile to save
Preventing falling when on ground continued

Hit Compile
Invoke onGround() by right clicking on Pengu to see that what you did works!

The boolean should return true if Pengu is on ground and false otherwise.
public boolean onGround()
{
    Object under = getOneObjectAtOffset(0, getImage().getHeight()/2-8, null);
    return under != null;
}
Preventing falling when on ground continued

public class Pengu extends Mover {

    public void act() {
        checkKeys();
        checkFall();
    }

    private void checkKeys() {
        if (Greenfoot.isKeyDown("left") ) {
            setImage("pengu-left.png");
            moveLeft();
        }
        if (Greenfoot.isKeyDown("right") ) {
            setImage("pengu-right.png");
            moveRight();
        }
    }

    private void checkFall() {
        if (onGround()) {
            setVSpeed(0);
        } else {
            fall();
        }
    }
}

Now we need to create code that will result in our character falling only if we are not on ground

Open the Pengu class

Delete the method fall() and replace with a new method called checkFall()

Implement the method checkFall in the body of the code

If the character is on ground, this stops the fall as vertical speed will be set to zero else the character will fall

Hit Compile and run to test!
Pengu falls!
Now we need to create code that will let our character jump!

First we need to add a key that will control the jump

Open the Pengu class and add this code

Pengu will jump when you press the space key if he is on ground
Getting Pengu to Jump Continued!

```java
public class Pengu extends Mover {
    public void act() {
        checkKeys();
        checkFall();
    }

    private void checkKeys() {
        if (Greenfoot.isKeyDown("left")) {
            setImage("pengu-left.png");
            moveLeft();
        }
        if (Greenfoot.isKeyDown("right")) {
            setImage("pengu-right.png");
            moveRight();
        }
        if (Greenfoot.isKeyDown("space")) {
            if (onGround()) {
                jump();
            }
        }
    }

    private void jump() {
        setVSpeed(-16);
        fall();
    }
}
```

Now we need to implement the jump method. Jumping is similar to falling but with upwards movement.

So we start with a negative vertical speed because the vertical speed will decrease until standstill.

Open the Pengu class and add this code:

Hit Compile and run to test it out!
public class Pengu extends Mover {
    private static final int jumpStrength = 16;
    public void act() {
        checkKeys();
        checkFall();
    }
    private void checkKeys() {
        if (Greenfoot.isKeyDown("left") )
        {
            setImage("pengu-left.png");
            moveLeft();
        }
        if (Greenfoot.isKeyDown("right") )
        {
            setImage("pengu-right.png");
            moveRight();
        }
        if (Greenfoot.isKeyDown("space") )
        {
            if (onGround())
                jump();
        }
    }
    private void jump() {
        setVSpeed(-jumpStrength);
        fall();
    }
}

Now let's use a variable in the Pengu class for the vertical speed when jumping because this will make it easier to find and change later!

Let's call it jumpStrength

Don't forget to also update the jump method with the new variable name!

Test out different jumpStrengths by changing the value.

Compile and run!
Running, falling, jumping!

Pengu should now be able to run, fall and jump correctly!
Creating a moving platform

We need a way for Pengu to get from side to side without falling!

We can create a moving platform that Pengu can surf on like the cloud!!
Moving Platforms!

import greenfoot.*;  // (World, Actor, GreenfootImage, and Greenfoot)

/**
* A cloud that moves back and forth between two defined points.
*/
public class Cloud extends Actor {

    private int speed = 4;

    /**
     * Move in the direction we are currently moving in.
     * Turn if we reach a turning point.
     */
    public void act()
    {
        setLocation ( getX() + speed, getY() );

        Actor actor = getOneIntersectingObject(null);
        if(actor != null) {
            actor.setLocation ( actor.getX() + speed, actor.getY() );
        }
    }
}
public class Cloud extends Actor
{
    private int speed = 4;
    private int leftTurn = 270;
    private int rightTurn = 480;

    /**
     * Move in the direction we are currently moving in. Turn if we reach a turning point.
     */
    public void act()
    {
        setLocation ( getX() + speed, getY() );

        Actor actor = getOneIntersectingObject(null);
        if(actor != null) {
            actor setLocation ( actor.getX() + speed, actor.getY() );
        }

        if (atTurningPoint()) {
            speed = -speed;
        }
    }

    /**
     * Test if we are at one of the turning points.
     */
    public boolean atTurningPoint()
    {
        return (getX() <= leftTurn || getX() >= rightTurn);
    }
}
Now Pengu can surf the cloud!
public boolean onGround() {
    Object under = getOneObjectAtOffset(0, getImage().getHeight()/2-8, null);
    return under != null;
}

public void setVSpeed(int speed) {
    vSpeed = speed;
}

public void fall() {
    setLocation( getX(), getY() + vSpeed);
    vSpeed = vSpeed + acceleration;
    if ( atBottom() )
        gameEnd();
}

private boolean atBottom() {
    return getY() >= getWorld().getHeight() - 2;
}

private void gameEnd() {
    Greenfoot.stop();
}

We want our game to be over if our character falls off and dies!

Open the Mover class

Create the method atBottom. This will decide if Pengu has fallen

Create the method gameEnd. This stops running Greenfoot

Call these methods in the fall method