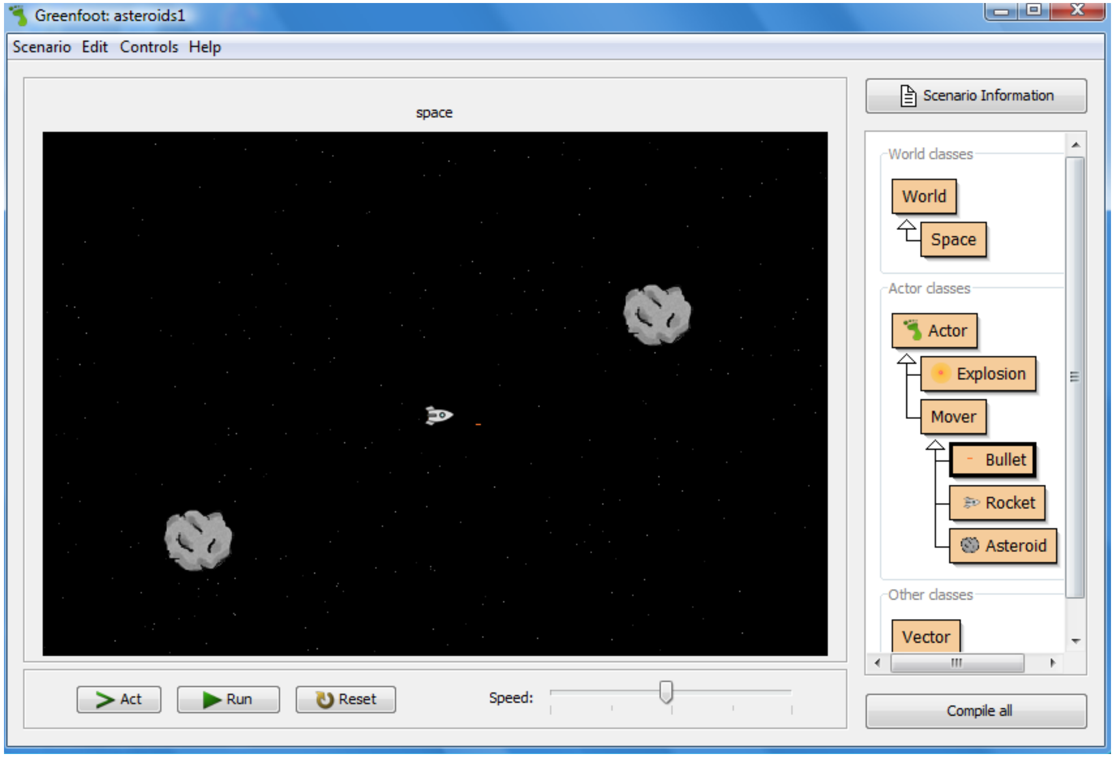
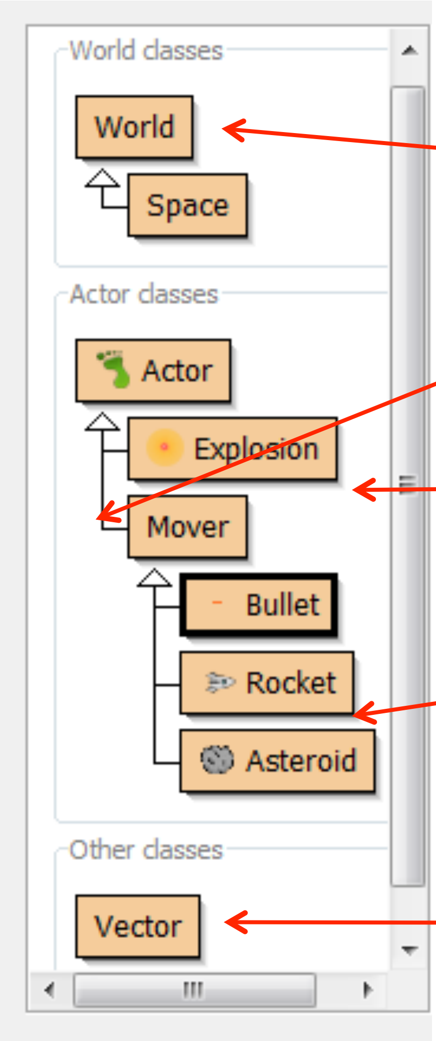
Chapter 1 - Getting to know Greenfoot (Asteroids exercises)

Original slides by Bruce Chittenden
Exercise edits by Scott Blanck

Greenfoot Second Example – Open book scenarios – chapter01 – asteroids1



1.8 Understanding the Class Diagram



World Class is always there in Greenfoot scenarios, it is built-in. Space represents a specific world for this scenario

Arrows show relationships

Explosion and Mover are subclasses of Actor

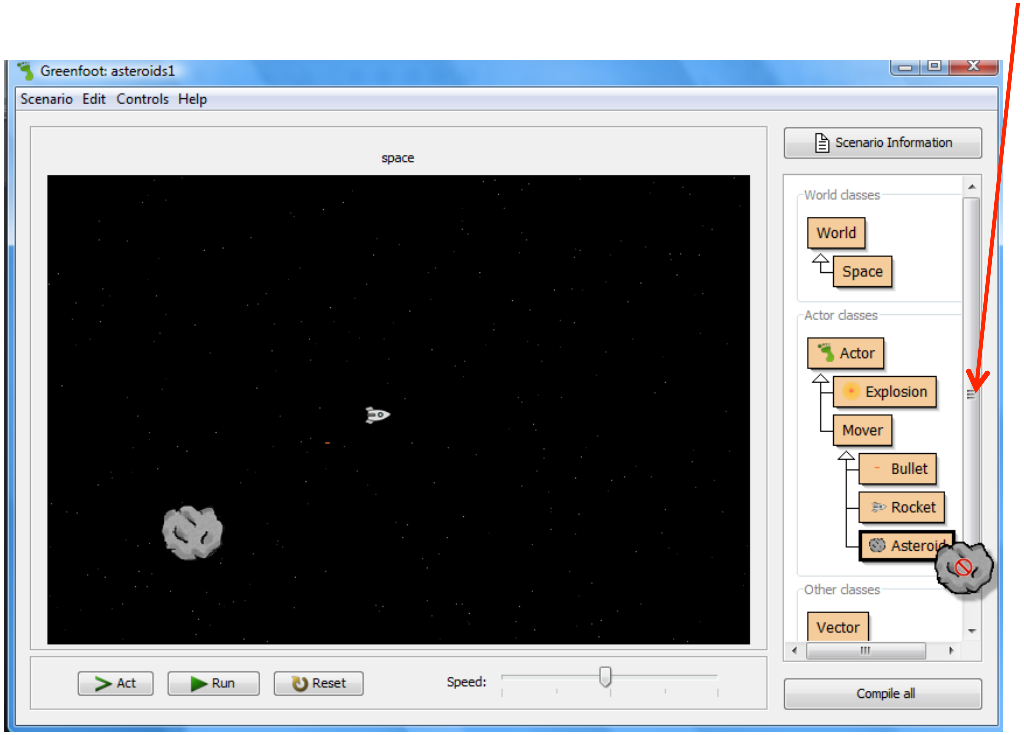
Bullet, Rocket, and Asteroid are subclasses of Mover.

Vector is a helper class

1.9 Playing with Asteroids

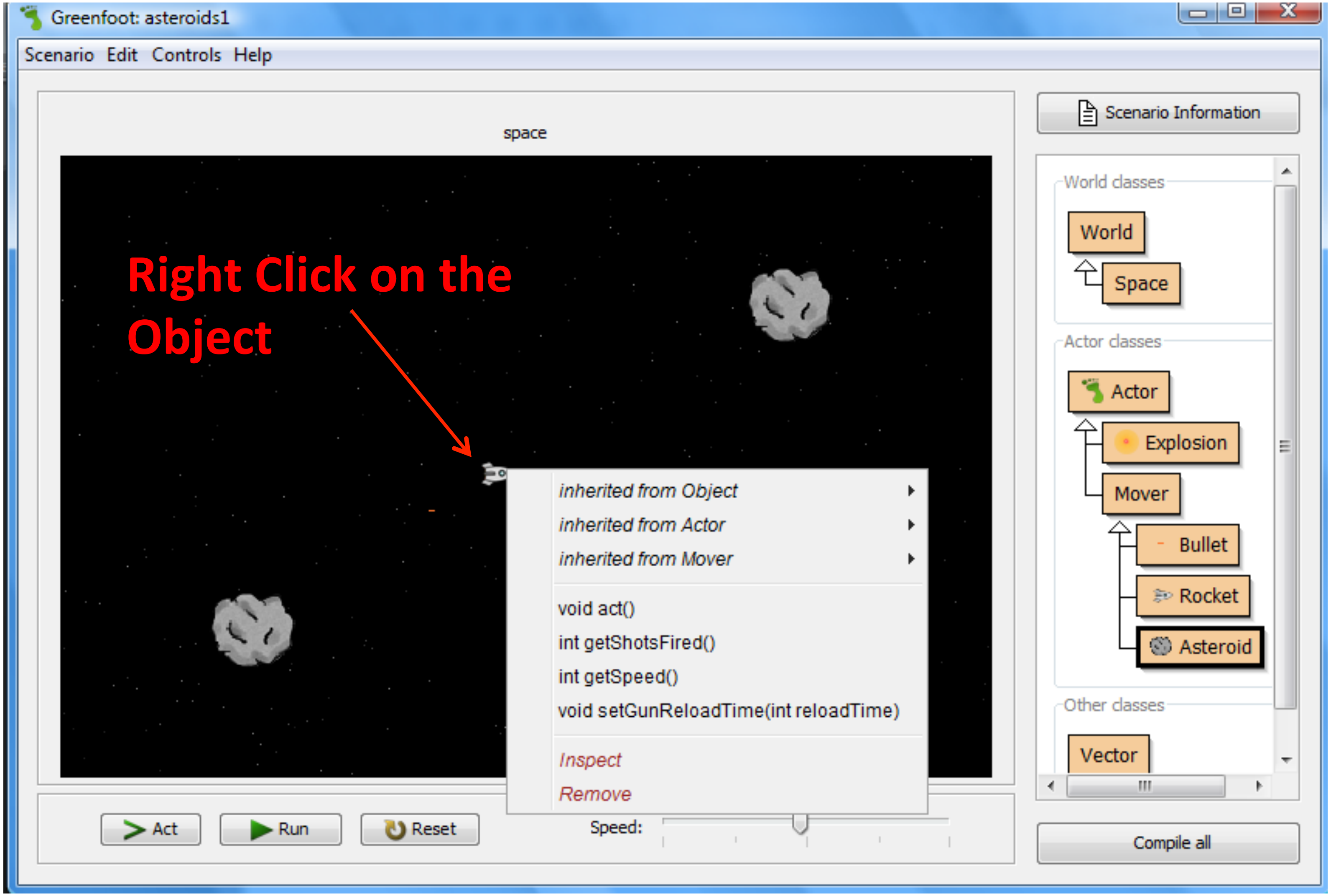
Start Playing by Creating Some Actor Objects (Objects of the Subclass of Actor).

Create Objects for Rocket, Bullet, Asteroid, and Explosion. What happens?



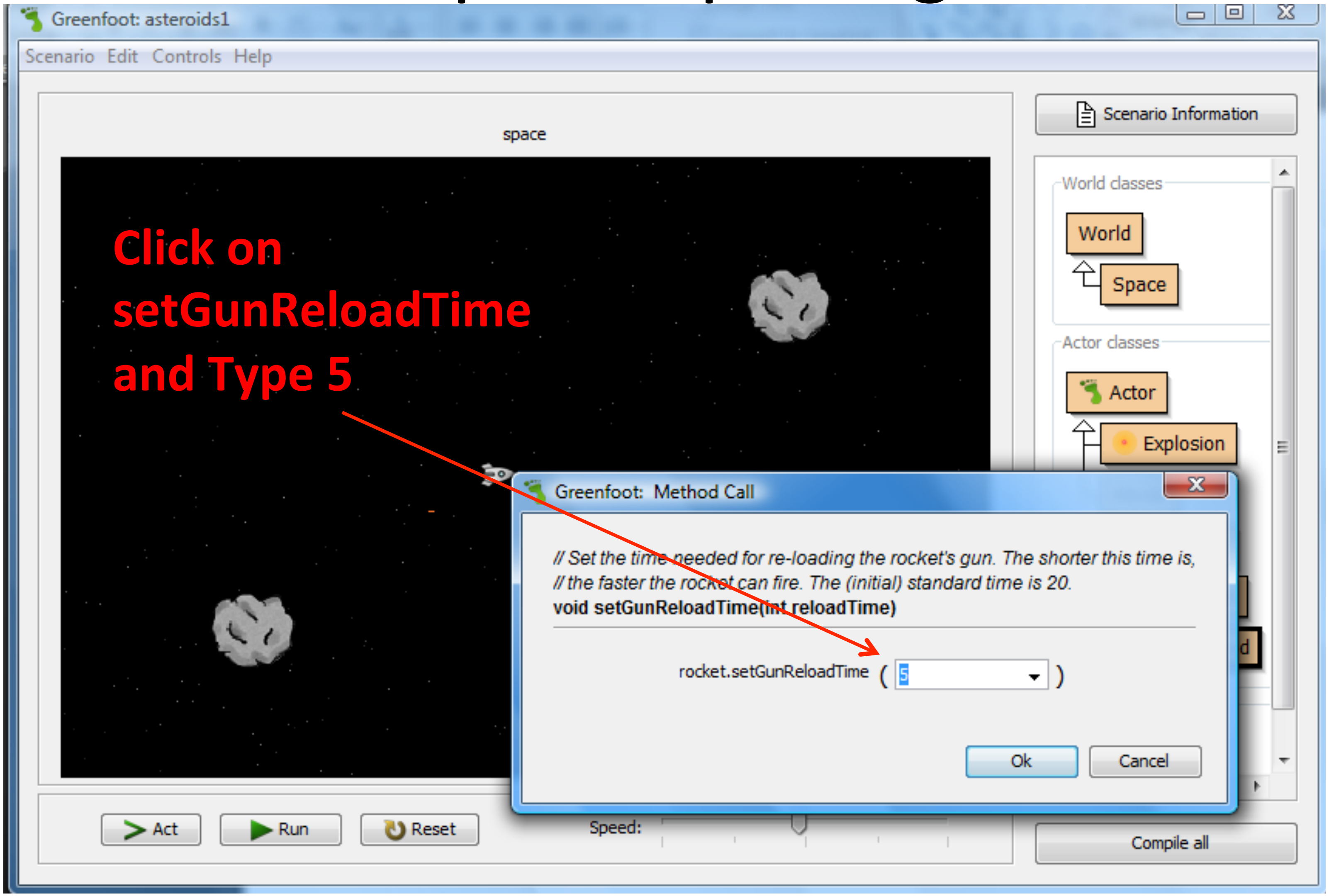
Exercise 1.10

Speed up firing with setGunReloadTime() method



Right Click on the Object

Exercise 1.10
Speed up firing



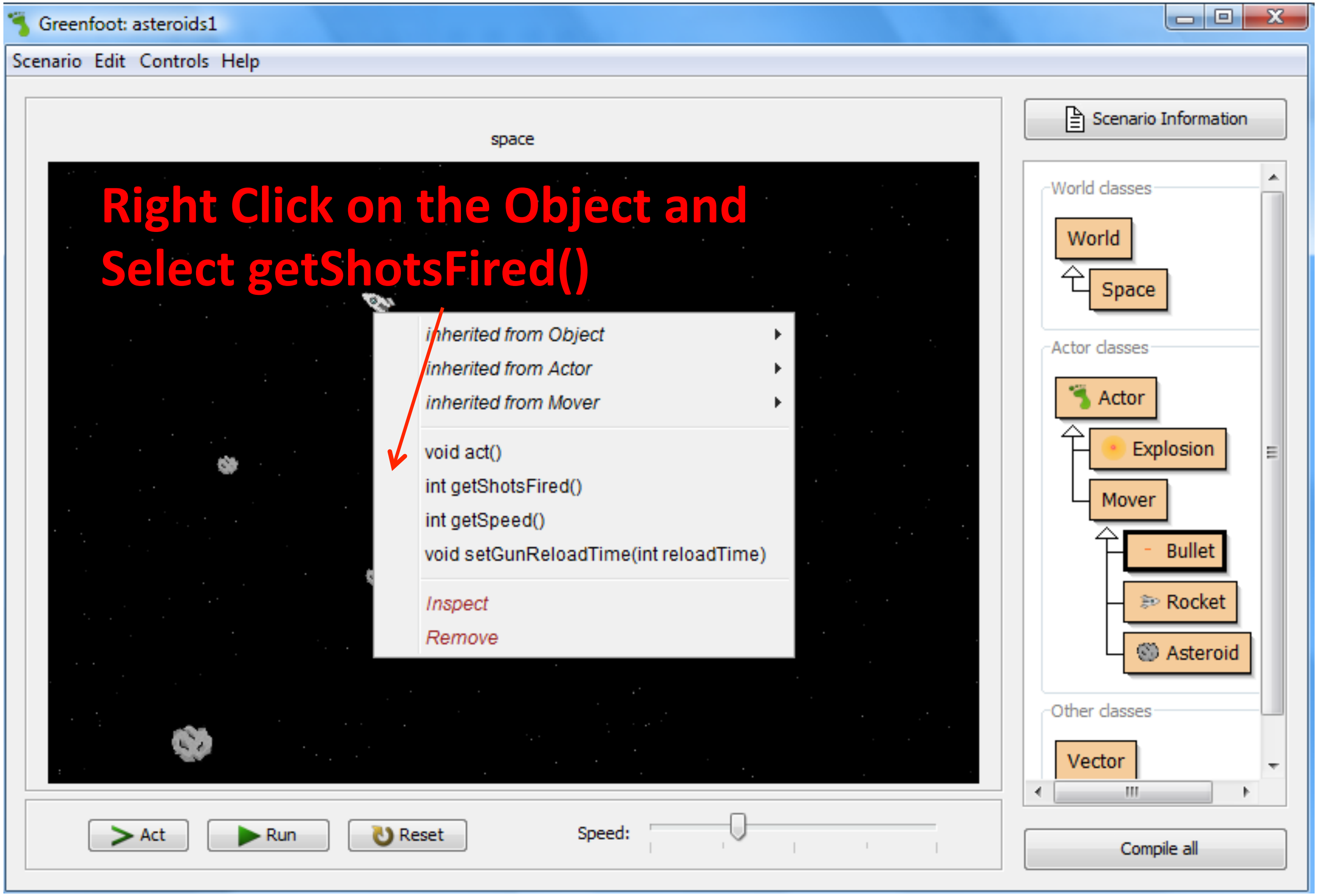
Click on setGunReloadTime and Type 5

Greenfoot: Method Call

// Set the time needed for re-loading the rocket's gun. The shorter this time is,
// the faster the rocket can fire. The (initial) standard time is 20.
void setGunReloadTime(int reloadTime)

rocket.setGunReloadTime (5)

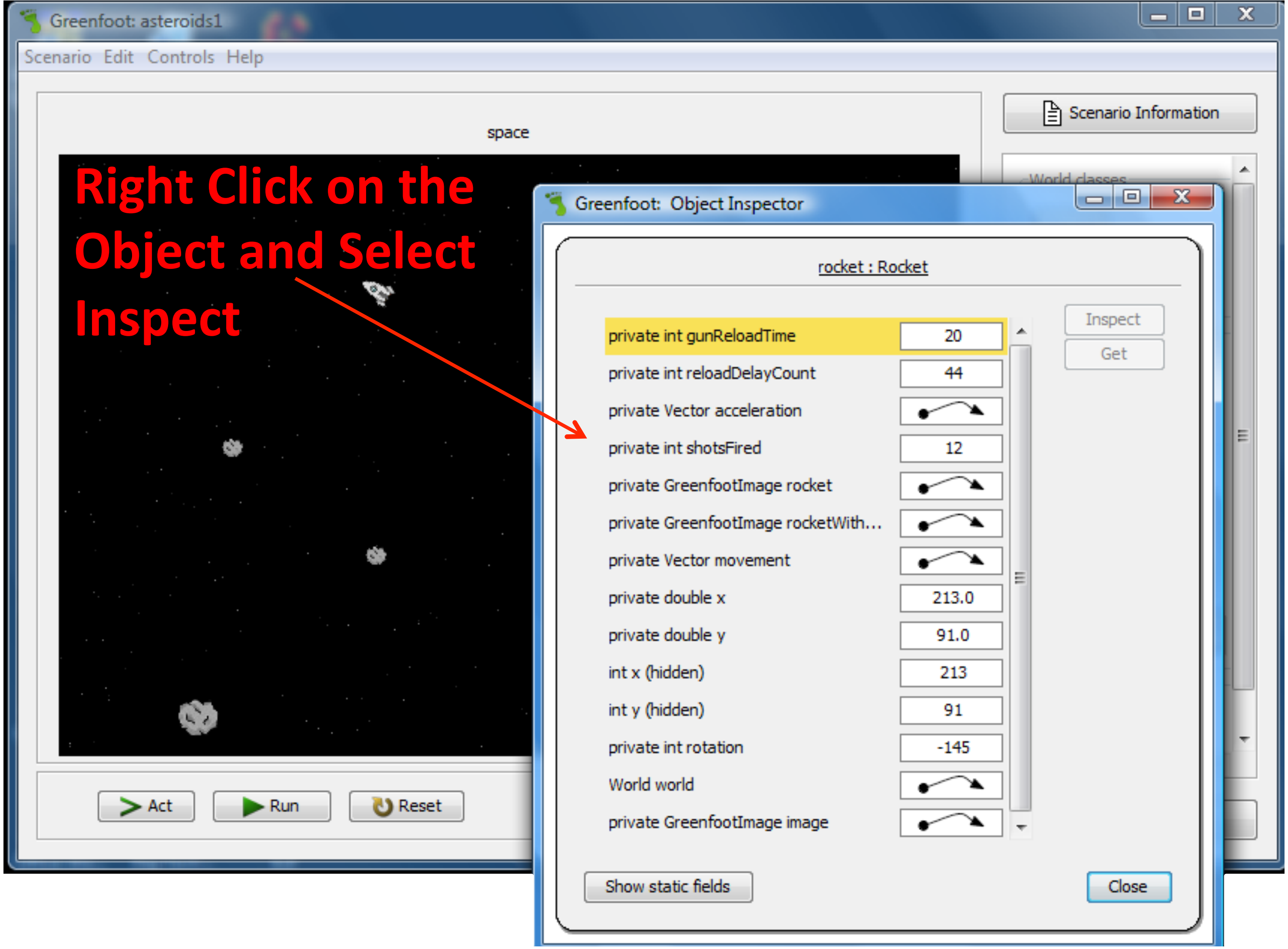
Exercise 1.11 – Destroy some asteroids. How many shots have you fired? Who can destroy two asteroids with the fewest shots?



Right Click on the Object and Select getShotsFired()

Exercise 1.11

See the state of an object



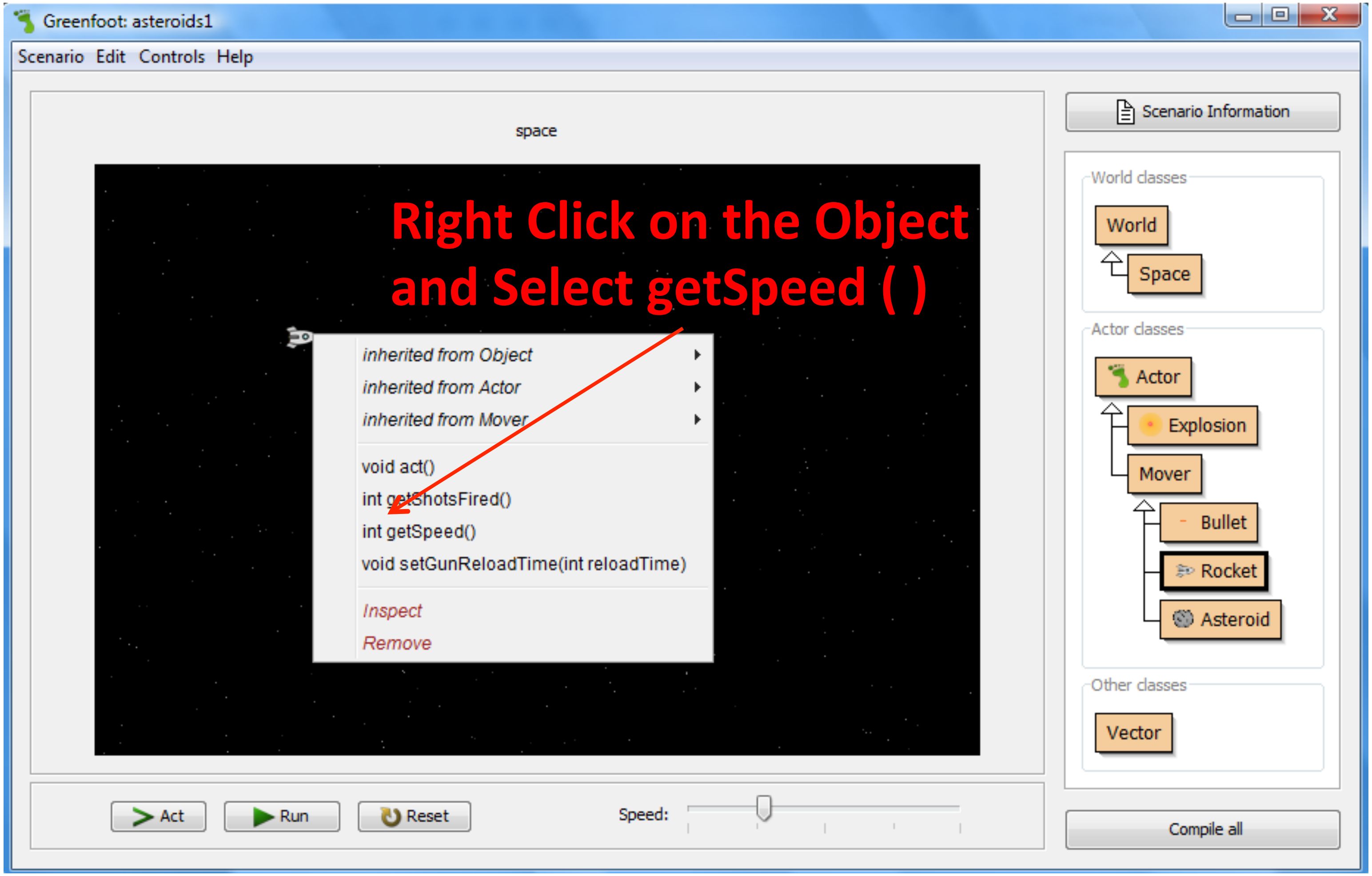
Right Click on the Object and Select Inspect

rocket : Rocket

private int gunReloadTime	20
private int reloadDelayCount	44
private Vector acceleration	→
private int shotsFired	12
private GreenfootImage rocket	→
private GreenfootImage rocketWith...	→
private Vector movement	→
private double x	213.0
private double y	91.0
int x (hidden)	213
int y (hidden)	91
private int rotation	-145
World world	→
private GreenfootImage image	→

Exercise 1.12

When you create a rocket, what is the initial speed?

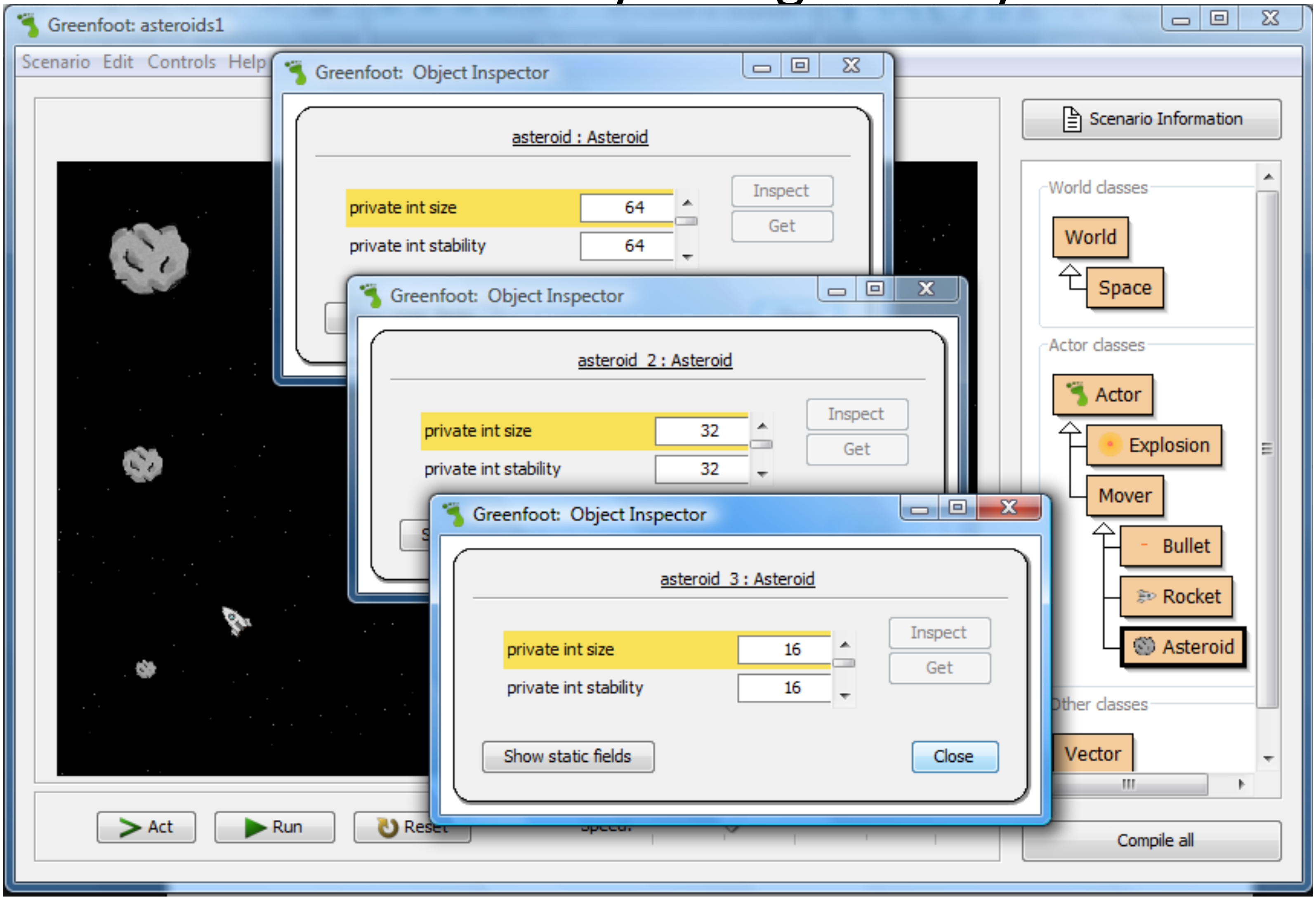


Right Click on the Object and Select getSpeed ( )

Exercise 1.13 – Asteroid stability

What is the *stability* of an asteroid after you place it?

How much does stability change after you shoot it?



Greenfoot: asteroids1

Greenfoot: Object Inspector

asteroid : Asteroid

private int size 64

private int stability 64

Greenfoot: Object Inspector

asteroid_2 : Asteroid

private int size 32

private int stability 32

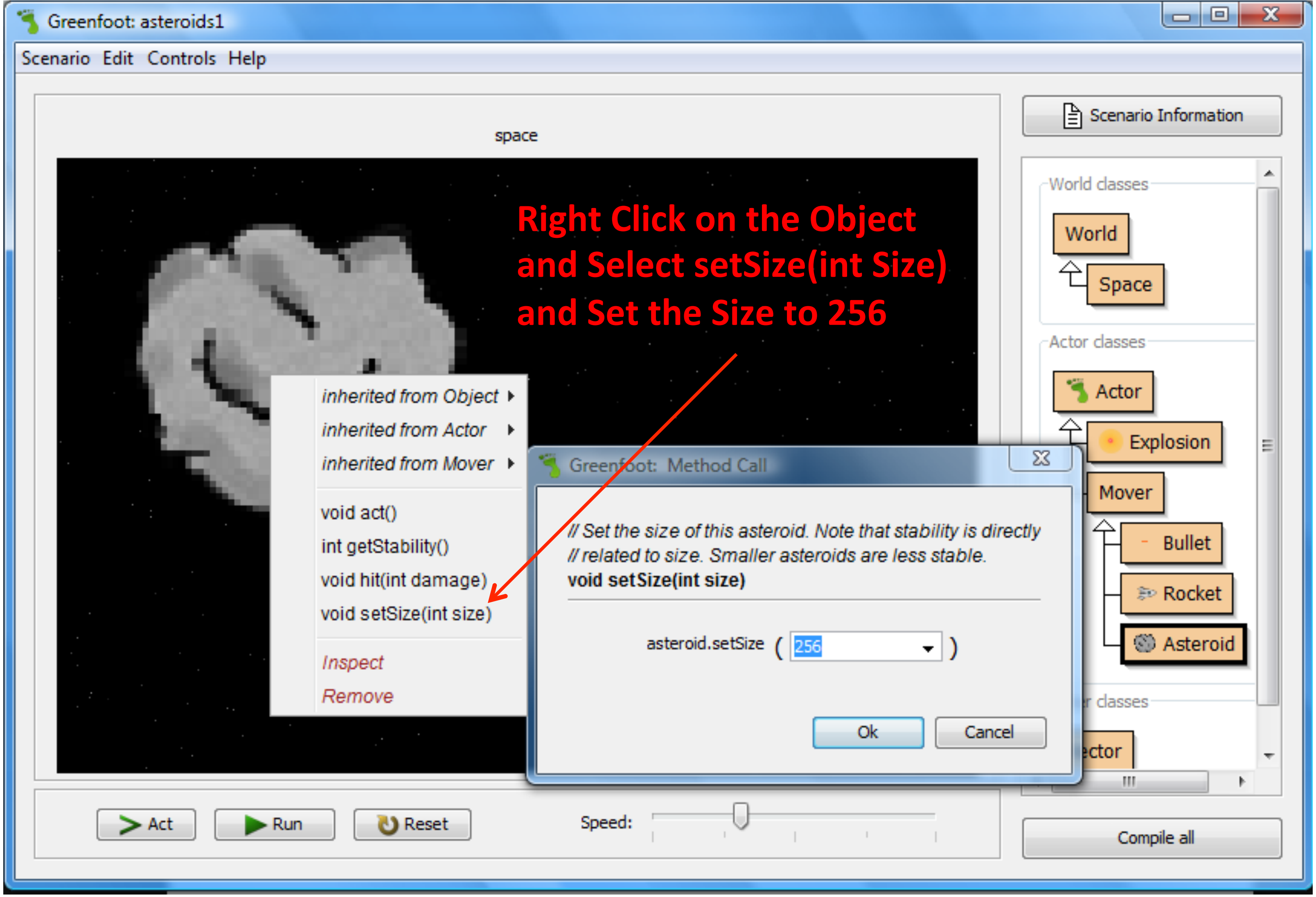
Greenfoot: Object Inspector

asteroid_3 : Asteroid

private int size 16

private int stability 16

Exercise 1.14
Make a very big asteroid.

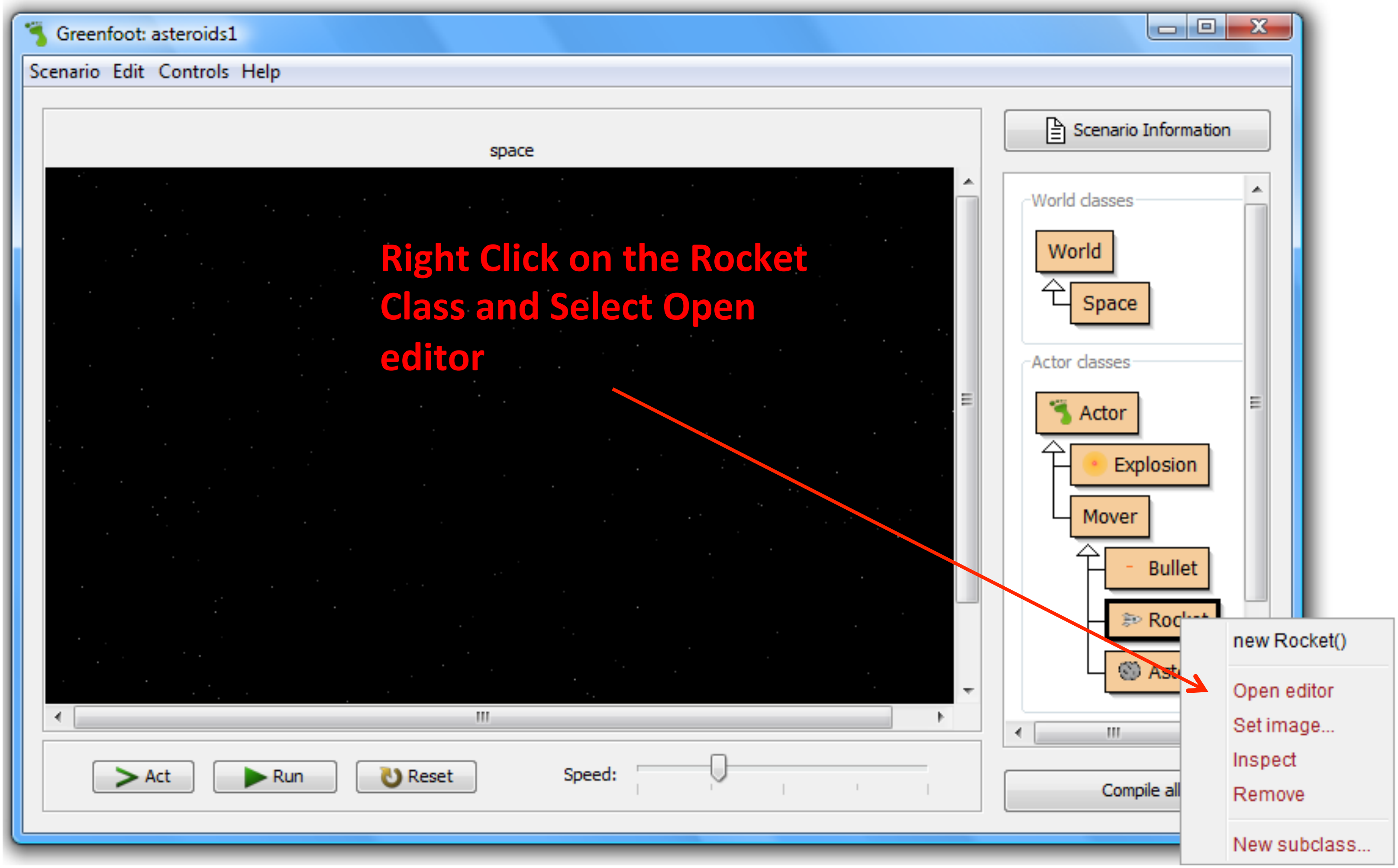


Right Click on the Object and Select setSize(int Size) and Set the Size to 256

// Set the size of this asteroid. Note that stability is directly
// related to size. Smaller asteroids are less stable.
void setSize(int size)

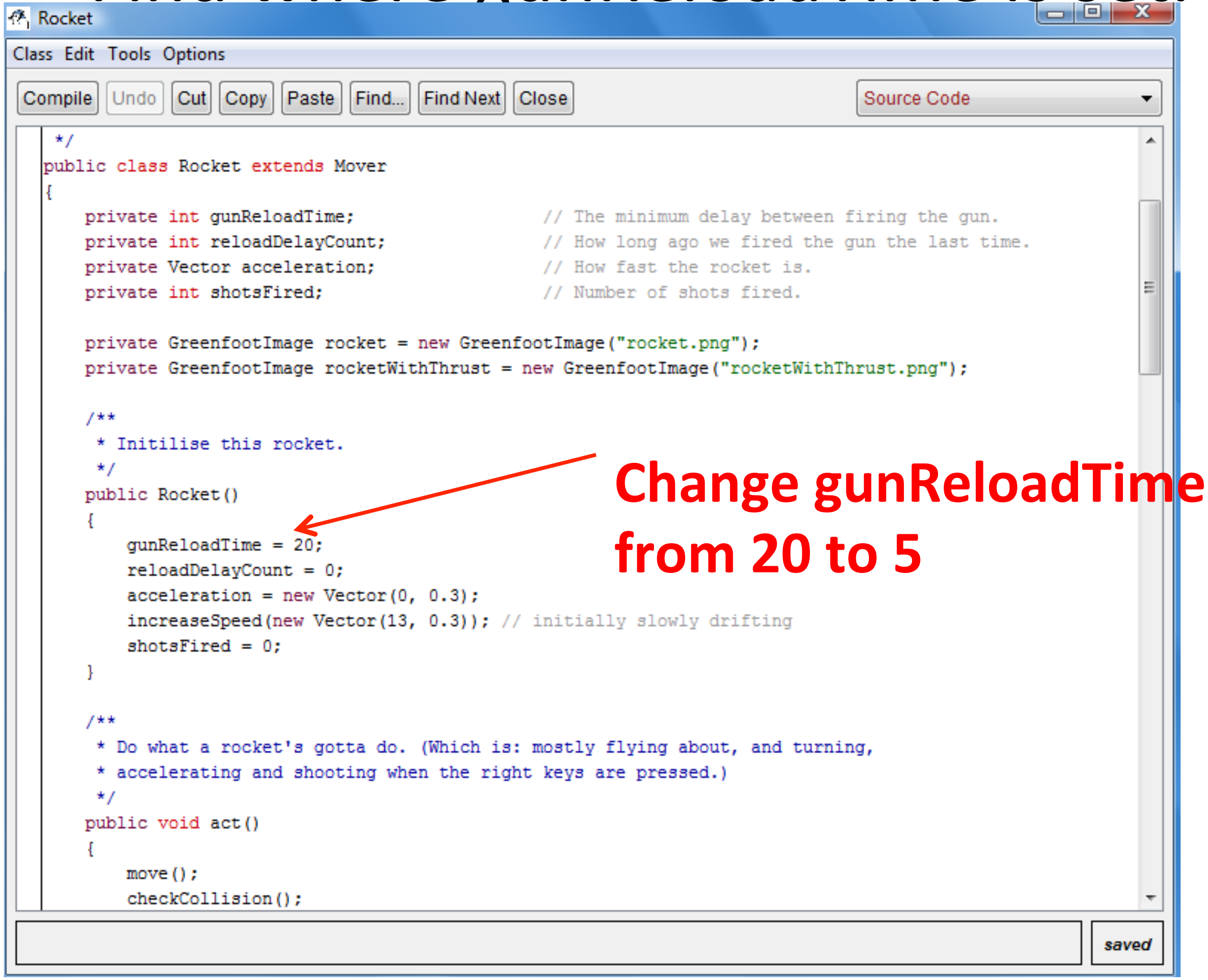
asteroid.setSize (256)

1.10 Open the Java Source Code for Rocket (or any object)



Right Click on the Rocket Class and Select Open editor

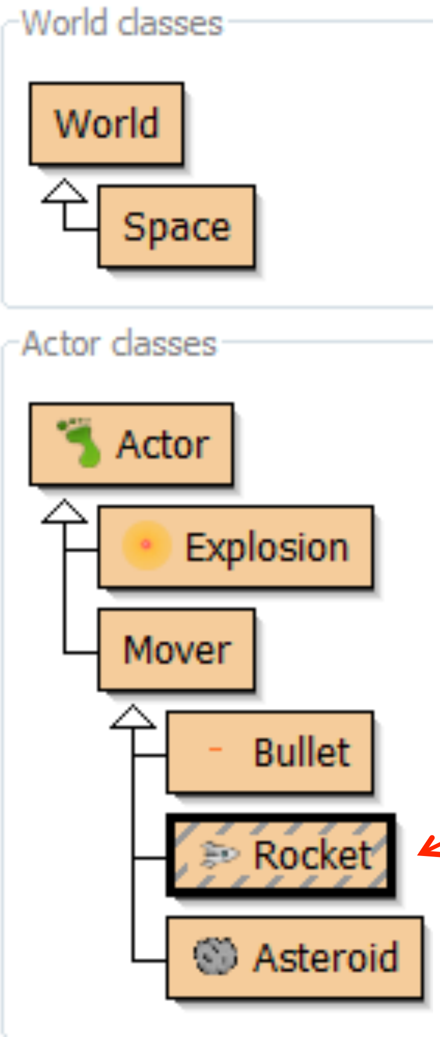
Source Code for Rocket –
Let’s change the source code to speed up firing.
Find where gunReloadTime is set.



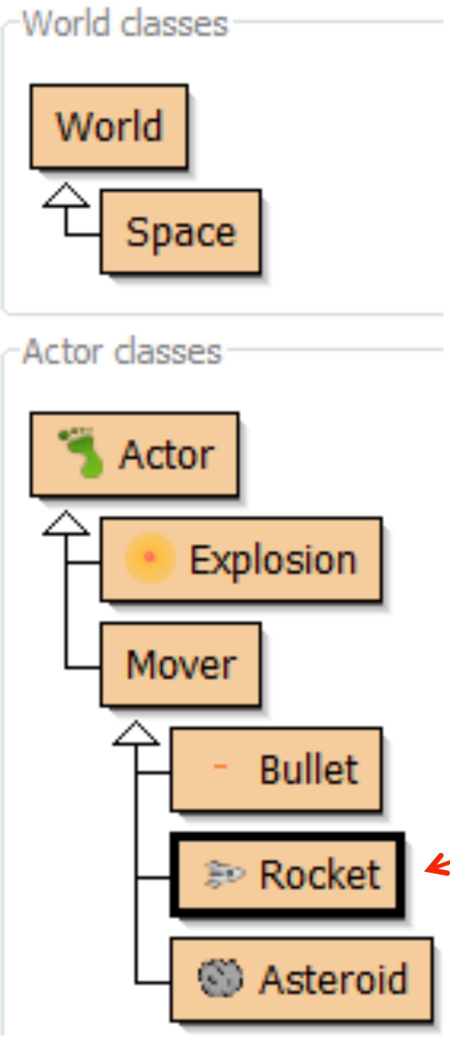
```
 */
public class Rocket extends Mover
{
    private int gunReloadTime;                  // The minimum delay between firing the gun.
    private int reloadDelayCount;               // How long ago we fired the gun the last time.
    private Vector acceleration;                // How fast the rocket is.
    private int shotsFired;                     // Number of shots fired.

    private GreenfootImage rocket = new GreenfootImage("rocket.png");
    private GreenfootImage rocketWithThrust = new GreenfootImage("rocketWithThrust.png");

    /**
     * Initilise this rocket.
     */
    public Rocket()
    {
        gunReloadTime = 20;
        reloadDelayCount = 0;
        acceleration = new Vector(0, 0.3);
        increaseSpeed(new Vector(13, 0.3)); // initially slowly drifting
        shotsFired = 0;
    }

    /**
     * Do what a rocket's gotta do. (Which is: mostly flying about, and turning,
     * accelerating and shooting when the right keys are pressed.)
     */
    public void act()
    {
        move();
        checkCollision();
```

Change gunReloadTime from 20 to 5

Exercise 1.15 – Notice the diagonal lines in classes. Must compile, then play with new firing.



Class Changed



Class Compiled

1.11 Summary

In this chapter, we have seen what Greenfoot scenarios can look like and how to interact with them.

- We have seen how to create objects and how to communicate with these objects by invoking their methods.
- Some methods are commands to objects, while other methods return information about the object.
- Parameters are used to provide additional information to methods, while return values pass information back to the caller.