**Swarming Robots: Cheat-Sheet**

Page break image

Default Greenfoot and Java methods:

|  |  |
| --- | --- |
| Code | What it does |
| **Greenfoot and Java methods:**  These code examples are native to Greenfoot and Java. | |
| Actor actor1 = new Actor(); | Creates a new Actor object named “actor1”. |
| addObject(actor1, 0, 0); | Adds the Actor “actor1” to the world with a x position of 0 and a y position of 0. |
| actor1.setRotation(180); | Makes actor1 face the left of the screen. Rotation goes from 0 to 359 starting from facing towards the right of the screen. |
| Greenfoot.getRandomNumber(100); | Returns a random number between 0 and 99. |
| variable++; | Increments the variable counter, meaning the value of variable is increased by 2. |
| for (int i=0; i<10; i++)  {  //Do code  } | A for loop. Loops 10 times, from i=0 to i=9. |
| if (x > 5)  {  //Do 1st code  }  else  {  //Do 2nd code  } | If else statement. Runs the 1st code if the condition x>5 is true, otherwise runs the second code. |
| setImage(“image2.png”); | Sets the image of the actor to “image2.png”. |
| move(5); | Moves the actor 5 units in the direction it is facing. |
| int[] intArray= {30, 55, 10}; | Creates an integer array with 3 items: 30, 55, and 10. |
| Greenfoot.playSound(“sound.mp3”); | Plays the sound file “sound.mp3”. |
| **SwarmRobot methods:**  Swarm Robot is a parent class of all swarm robots. | |
| bounceOffEdge(); | If the actor is at the edge of the world then it will turn around and bounce of the edge. |
| loopThroughEdge(); | If the actor is at the edge of the world then it should appear on the opposite side of the world. |
| **FireflySuper methods:**  Used to abstract complexities so that the Firefly algorithm can be implemented easily with a Firefly subclass | |
| crawlAround(); | Mimics the way an insect crawls. Moves the actor forward and turns it randomly. |
| neighbourFlyFlashing(1000); | Returns true if a neighbour firefly is flashing within 1000 cells. |
| **BoidSuper methods:**  Used to abstract complexities so that the Boid flocking algorithm can be implemented easily with a Boidsubclass | |
| alignment(); | Returns the heading that is the average of the neighbours heading. |
| cohesion(); | Returns the heading towards the centre of the neighbours. |
| separation(50); | Returns the heading needed to move away from other neighbours. The boid looks for neighbours to move away from in a distance of 50 cells. |
| averageOfAngles(angleArray); | Returns the average angle of all the angles in an array. |
| turnSlightly(x); | Turns the actor towards the angle x by only 1 degree. The actor will turn the shortest way towards the angle. |