**GridWorld Object Reference Document**

GridWorld is a graphical simulation program that runs off a matrix. Objects with specific behaviors are place within the “Grid” and the simulation evolves over a series of steps, with different “Actors” interacting with one another. There are several core classes that Grid is based upon:

**Location Class** – The Grid is essentially a 2D array of Locations. Each Location has the following methods:

**public Location (int r, int c)** – Constructor

**public int getRow()** – returns the row within the grid that this location is found at

**public int getCol()** – returns the column within the grid that this location is found at

**public Location get AdjacentLocation(int direction)** – Returns the Location object found in the received direction

**public int getDirectionToward(Location target)** – Returns the int direction from this Location to get to target Location

**public boolean equals(Object other)** – Returns true if the received object is the same Location

**public String toString()** – Outputs the Location object

The Location Class also has several constants:

**Compass Directions**

**NORTHWEST = 315 NORTH = 0 NORTHEAST = 45**

**WEST = 270 EAST = 90**

**SOUTHWEST = 225 SOUTH = 180 SOUTHEAST = 135**

**Turn Angles**

**LEFT = -90 RIGHT = 90 HALF\_LEFT = -45 HALF\_RIGHT = 45**

**HALF\_CIRCLE = 180 AHEAD = 0 FULL\_CIRCLE = 360**

**Grid Class (Interface)** – This is the 2D array that holds all the Locations and Actors within the simulation

**public int getNumRows()** – Returns the number of rows in the Grid

**public int getNumCols()** – Returns the number of columns in the Grid

**public boolean isValid(Location loc)** – Returns true if that is a valid Location within the Grid, otherwise false

**put(Location loc, Object E)** – Puts an object E at Location loc within the grid.

**remove(Location loc)** – Remove the object that exists at the specified Location from the Grid

**get(Location loc)** – Gets the object that is located at Location loc within the Grid.

**public ArrayList<Location> getOccupiedLocations()** – Returns an ArrayList of all Locations in the Grid containing Objects

**public ArrayList<Location> getValidAdjacentLocations()** – Returns an ArrayList of all valid Locations within 1 move

**public ArrayList<Location> getEmptyAdjacentLocations()** – Returns an ArrayList of adjacent Locations with no objects

**public ArrayList<Location> getOccupiedAdjacentLocations()** – Returns an ArrayList of all adjacent Locations with objects

**public ArrayList<Object> getNeighbors(Location loc)** – Returns an ArrayList of all Objects adjacent to Location loc.

**Actor Class** – This is the superclass that all objects used in the simulation will inherit

**public Actor()** – No-argument constructor – We won’t instantiate any Actor objects, instead their subclasses

**public Color getColor()** – Returns the color of the Actor

**public void setColor(Color newColor)** – Changes the color of the Actor to newColor

**public int getDirection()** – Returns the direction the Actor is facing as an integer

**public void setDirection(int newDir)** – Changes the direction of the actor to newDir

**public Grid<Actor> getGrid()** – Gets the Grid of Actors that the Actor object is a part of

**public Location getLocation()** – Returns the Location in the Grid where the Actor is found

**public void putSelfInGrid(Grid<Actor> gr, Location loc)** – Puts that Actor into the Grid at Location loc

**public void removeSelfFromGrid()** – Removes the Actor from the Grid

**public void moveTo(Location loc)** – Moves the Actor to that new Location

**public void act()** – This method is called each “Step” of the simulation

**public String toString()** – This method outputs the Actor object.

**Bug Class (Extends Actor)** – A bug is an Actor that moves around the simulation, leaving Flowers, but mostly gentle

**public Bug()** – No-arg Constructor

**public Bug(Color color)** – Constructor that assigns the received color to the Bug instance

**public void act()** – This method would override the act method received from the Actor Class

**public void turn()** – Bug turns clockwise 45 degrees

**public void move()** – Bug moves 1 square in whatever direction he’s facing

**public boolean canMove()** – Returns true if it can move in the direction its facing, otherwise false

**Critter Class (Extends Actor)** – A critter is an Actor that will often remove other objects from the Grid if it gets near them

**public void act()** – Inherited method that is often overridden from Actor class

**public ArrayList<Actor> getActors()** – Returns an ArrayList of all Actors in the Grid

**public void processActors(ArrayList<Actor> actors)** – All actors in the received ArrayList are “acted upon” by Critter

**public ArrayList<Location> getMoveLocations()** – returns an ArrayList of Locations to which the Critter can move

**public Location selectMoveLocation(ArrayList<Location> locs)** – Returns the one Location to move to from the ArrayList

**public void makeMove(Location loc)** – Actually moves the Critter to the received Location

**Rock class (Extends Actor)** – A rock is essentially non-interactive by default

**public Rock()** – No-arg constructor

**public Rock(Color color)** – Constructor that assigns initial color of the Rock

**public void act()**

**Flower Class(Extends Actor)** – Flowers are left behind by Bugs (by default) and removed by Critters (by default)

**public Flower()** – No-arg constructor

**public Flower(Color color)** – Constructor that assigns initial color of the Flower

**public void act()**