Basic rules for assignment A5

Work in groups of 1 or 2. A5 may be done with one other person. The A3 handout contains instructions for grouping and also talks about how group members should work together. Those instructions apply to A5 as well.

Academic integrity. Never look at or be in possession of the assignment A5 code of another group, in this semester or a previous one. Do not show or give your code to another student. Do not post assignment A5 code on the Piazza or elsewhere. If our system finds that two solutions are very similar, expect that we will ask you and the other person to meet with us and explain what happened.

Naturally, you may discuss A5 with others, but the discussion should not extend to writing actual code, picking variable names, agreeing on specifications or comments, etc. We check for unexpected similarities and consider it to be a violation of the academic integrity rules if code was shared.

What A5 is about

We have studied in depth the genetic makeup of a number of unusual and intriguing creatures. Now, we travel back in time, to revisit the story of how these particular species were first discovered by science. In this assignment, you will write a program to guide "the naturalist", who appears to the left, and who boldly explores an uncharted tropical island and collects specimens of local wildlife. (Naturally, the naturalist could be a woman or a man. Read carefully, and you will see that our discussion below is not gender biased.)

The naturalist begins on a ship, moored just off shore. Their goal is to collect all the animals on the island and bring them to the ship. Only 3 animals can be carried at a time, so this will require many trips to the ship.

The map is represented as a graph, with each square represented as a node. A node may be water; land with an obstacle, like a tree; or a place to which the naturalist can move, which may or may not have animals. The naturalist does not have a global view of the map but can see and move to adjacent nodes. The naturalist can search their current location for animals and, having found them, can pick them up and drop them.

To the right is part of one possible island that the naturalist may land on and explore. The piece of land jutting out to the right leads to another part of the island, with more animals. The ship is on the north. The naturalist has left the ship and is wandering about. Naturally, the naturalist can land on many different islands, with different shapes, trees, and animals. So the program you write will work with any such island.

Earning/losing points

The ultimate goal is to collect all animals and drop them on the ship. A naturalist is extremely competitive and wants to capture as many animals in as short a time as possible, but remember that the naturalist can hold only 3 animals at a time. Try to pick up more, and they are all dropped immediately.

The naturalist earns or loses points for various actions:

- **Exploration:** Moving to a node for the first time, 10 points.
- **Retrieval:** Dropping one animal on the ship: 100 points.
- **Mission accomplished:** Dropping all animals on the ship: 1000 point bonus points.
- **Movement:** Moving to a node: *Costs* 1 point + the number of animals being carried. Example: moving while carrying three animals costs four points.

Competition

Three prizes awarded for outstanding solutions:

- Highest score
- Fastest clock time
- Most innovative strategy

The competition will be run on new maps, larger and with more animals. To enter the competition, please say so in your README file. Tell us which category. Getting both the highest score and fastest clock time is probably not possible.

Grade for the assignment

There are several steps to this program. A solution that has the navigator visit all nodes is the minimum that should be accomplished. If this solution is well documented and well written, it will receive a grade of 70.

A solution that brings all the animals back to the ship with a positive score will receive a grade of 100 —if it is well documented and well written.

There are gradations between these two, which we can't explain yet. We will provide an update to the document on the Piazza at some point.

What we give you

This assignment is a departure from previous assignments, in that you will not need any of your old code. We give you, in a5.zip, a bunch of .class files (and the source file for one class), some images, and the API documentation that was created from the source. You will use the documentation to help you learn about the assignment and then write it. Here is some basic information:

Class Node. An instance of this class contains information about a node of the graph. Given a node n, you can, for example, tell whether it is the ship (n.isShip()), get its x- and y-coordinates, and determine whether it is adjacent to another node (n.isAdjacent(m).

Of extreme importance are methods setUserData(T data) and getUserData(). The naturalist, while moving around the island, visiting nodes, may to take notes. When walking around the island for the first time, the naturalist can only see local stuff, e.g. the animals at that node and the adjacent nodes. The notes the naturalist writes down will be placed in a mixture of fields and information that can be stored in the nodes.

You can create your own class NodeData (say) and have its instances contain the notes you need. You can then save the information, for example, using n.setUserData(**new** NodeData(…), and later you can retrieve it using, say, NodeData d= n.getNodeData. You design class NodeData yourself.

Class Naturalist. Class Naturalist contains methods that a naturalist uses to move around the island. The naturalist is always at a *current location*. From that location, the naturalist can

- obtain the node for the current location,
- obtain a list of adjacent nodes to which they can move,
- move to an adjacent location,
- obtain a list of animals at the location, collect an animal, and drop animals.

Class Naturalist has a constant, MAX_ANIMAL_CAPACITY, which is the maximum number of animals they can carry at one time. Currently, that value is 3, but we reserve the right to change it. In your program, *never* use the number 3 for the capacity; instead, use MAX_ANIMAL_CAPACITY.

Your subclass of abstract class Naturalist

Your task is to write a subclass of abstract class Naturalist. This is the only file that you will submit. If you need to declare any other classes, put them in this file, as a nested classes.

Your subclass must override abstract method run(). This method is called to start your naturalist off on their wanderings. When called, the naturalist is at the ship, and method run() must move the naturalist around, captur-

ing animals and bringing them back to the ship.

To help you understand how you might do this, we give you the source for class RandomWalkNaturalist, which implements a naturalist that has had too much to drink, so they wander around in a random manner.

Class Simulator

Method Simulator.main(String[] args) is called to start execution of the program. It reads parameter args (we discuss args below), initializes everything, creates an instance of your subclass of Naturalist, and calls its method run().

You specify the arguments to main in the Run Configuration file in Eclipse. The first argument *must* be the name of the subclass of naturalist that you write. The rest of the arguments are optional:

- --map=mapfile.txt Use this to load a map file, which givesthe format of the map. We may explain this later, but don't count on it
- --headless Run without a GUI
- --seed=N Use N as the seed for the random number generator

What to do for this assignment

We give no times for milestones for this assignment, leaving that up to you. But the worst thing you can do is wait until the day before the assignment to get started. A5 will require pondering on the best way to do things and on what information to save as the naturalist explores the island. It will take time to wrap your head around some of the issues. You may want to do some searching of the text or even the internet to find good algorithms to use. At some point, you may use any of the algorithms we have seen or even not seen —depth-first search, breadth-first search, minimum spanning tree, shortest paths, all-pairs shortest paths.

Step 1. Set up the program in Eclipse and run it. Download a5.zip and unzip it. Then do the following:

1. Start a new Eclipse project named, say, a5.
2. From the unzipped file, drag directories classes and game_tiles to the project in Package Explorer pane. When it asks, tell it to copy the files, not link to them.
3. From the unzipped file, drag file RandomWalkNaturalist.java to the src directory of the project.
4. The purpose of this step is to tell Eclipse to use the .class files in directory classes. With project a5 selected in the Project Explorer Pane, select menu item Project → Properties; in the window that opens, select Java Build Path / Libraries and then Add Class Folder; scroll down to the classes directory and check its box; hit OK, and the window disappears; click OK in the open one.
5. Put directory doc wherever you want. After putting it somewhere, open the directory and double-click on file index.html; that file will open in your favorite browser. This is the javadoc specifications that were abstract from the java source files. This is what you will look at to figure out how to use the various classes and their methods.

To run the program with a naturalist who wanders around randomly, do the following. With project a5 selected in the Package Explorer pane, select item Run → Run Configurations … In the window that opens, (1) Put the Name "assignment a5" (or whatever you want), (2) Put "A5" for the Project (or whatever you called the project), (3) Put "Simulator" for the Main class, (4) Put "RandomWalkNaturalist" for the Arguments. Click Apply and then Run.

Well, you can see the naturalist walking around aimlessly. You can speed him up or slow him down –move the slider all the way to the right and it continues to go, very fast, but seemingly forever. Stop execution by hitting the normal stop button. You can see one of Gries's solutions execute by choosing Run → Run Configuration and changing the argument to "NaturalistGries". Later, when you build your own subclass of Naturalist, change the arguments field in Run Configurations to the name of your subclass.

Step 2. Write the basic program. This assignment should probably be completed in stages, but we leave the design of those stages and their implementation to you. You know the naturalist has to pick up the animals and bring them back to the ship, but you know that trying to do that on a first traversal of the island can be ineffi-

cient. You know that information about the graph can be stored in each Node and in fields (static/non-static) of the subclass of Naturalist that you write. Work from that information.

Procedure run() should be fairly short, calling other methods to perform various actions. As an example, run() in NaturalistGries is 24 lines long, including the specification, some comments, some statements to calculate and print the time it takes to do a particular task. It also has two possible ways of calculating information based on notes taken, and it chooses between them based on a static field (this allowed to compare two different ways of doing things). All this in 24 lines, by making good use of other methods.

All methods must be well specified. A poorly documented program will not receive full credit.

Step 3. Submit on the CMS before the due date/time. The due date is Saturday, 4 May, at midnight. One point penalty will be given for each day late. No submissions after Tuesday, 7 May.

Prepare a readMe file (a txt file, Word file, or pdf file) that:

1. Says whether you want to enter your submission in the competition and in which categories.
2. Describes what stages your method run() performs and what algorithms you used at each stage (if you use some well-known graph algorithm).
3. Describes what you think of this assignment and the competition —and the whole course if you want.

Submit on the CMS your readMe file and your subclass of the Naturalist.