AP Computer Science A
GridWorld - Activity 3

Name _____
Date _____ Period _____

Activity:
A) Look at the following class files:

```
GridWorldCode/framework/info/gridworld/grid/Grid.java
GridWorldCode/framework/info/gridworld/grid/Location.java
```

Read: Pages 16 to 18 in the GridWorld Student Manual

Questions:
Consider these statements when answering the following questions.

```
Location loc1 = new Location(4, 3);
Location loc2 = new Location(3, 4);
```

1) How would you access the row value for `loc1`?

2) What is the value of `b` after the following statement is executed?

```
boolean b = loc1.equals(loc2);
```

3) What is the value of `loc3` after the following statement is executed?

```
Location loc3 = loc2.getAdjacentLocation(Location.SOUTH);
```

4) What is the value of `dir` after the following statement is executed?

```
int dir = loc1.getDirectionToward(new Location(6, 5));
```

5) How does the `getAdjacentLocation` method know which adjacent location to return?

Read: Pages 20 to 21 in the GridWorld Student Manual

Activity:
B) Open the API for the Grid interface in your **web browser**.

```
GridWorldCode/javadoc/info/gridworld/grid/Grid.html
```

Questions:
6) How can you obtain a count of the objects in a grid? How can you obtain a count of the empty locations in a bounded grid?

7) How can you check if location (10,10) is in a grid?

8) `Grid` contains method declarations, but no code is supplied in the methods. Why? Where can you find the implementations of these methods?

9) All methods that return multiple objects return them in an `ArrayList`. Do you think it would be a better design to return the objects in an array? Explain your answer.

Read: Pages 22 to 23 in the GridWorld Student Manual

Activity:
C) Open the following class file in Geany and answer the questions below.

```
GridWorldCode/framework/info/gridworld/actor/Actor.java
```

Questions:
10) Name three properties of every actor.

11) When an actor is constructed, what is its direction and color?

12) Why do you think that the `Actor` class was created as a class instead of an interface?

13) Work with the `BoxBugRunner`. Can an actor put itself into a grid twice without first removing itself? Can an actor remove itself from a grid twice? Can an actor be placed into a grid, remove itself, and then put itself back? Try it out. What happens?

14) How can an actor turn 90 degrees to the right?

Read: Pages 24 to 25 in the GridWorld Student Manual

Activity:
D) Look at the following class files:

```
GridWorldCode/framework/info/gridworld/actor/Bug.java
GridWorldCode/framework/info/gridworld/actor/Flower.java
GridWorldCode/framework/info/gridworld/actor/Rock.java
```

Questions:
15) Which statement(s) in the `canMove` method ensures that a bug does not try to move out of its grid?

16) Which statement(s) in the `canMove` method determines that a bug will not walk into a rock?

17) Which methods of the `Grid` interface are invoked by the `canMove` method and why?

18) Which method of the `Location` class is invoked by the `canMove` method and why?

19) Which methods inherited from the `Actor` class are invoked in the `canMove` method?

20) What happens in the `move` method when the location immediately in front of the bug is out of the grid?

21) Is the variable `loc` needed in the `move` method, or could it be avoided by calling `getLocation()` multiple times?

22) Why do you think the flowers that are dropped by a bug have the same color as the bug?

23) When a bug removes itself from the grid, will it place a flower into its previous location?

24) Which statement(s) in the `move` method places the flower into the grid at the bug's previous location?

25) If a bug needs to turn 180 degrees, how many times should it call the `turn` method?

Activity:
E) Make a new directory **Activity3** inside the **GridWorldCode** directory (**GridWorldCode/Activity3**). Do all your work in this directory. (Note: This activity is different than what is mentioned in the GridWorld Student Manual.)

Create a new class called **Blossom** that extends the **Flower** class. The **Blossom** starts out green and gets darker just like a **Flower**. A **Blossom** has a lifetime, and at the end of its lifetime it is removed from the grid (**removeSelfFromGrid** method in the **Actor** API). Create two constructors for the **Blossom**. A no-args constructor that sets the lifetime to 10 (steps), and a second constructor with a parameter that specifies the lifetime.

Create a new class called **Jumper** that extends the **Bug** class. **Jumper** is the color blue. This actor can move forward two cells in each move, so it can "jump" over obstacles, like rocks and flowers (and blossoms). If there are no empty cells two away in its current direction, it tries new directions until it finds and moves to an empty cell two away. If it moves in a different direction, it faces in the direction it just moved. Instead of leaving a **Flower** like a **Bug** does, it leaves a **Blossom** when it moves.

Here are some situations left for you to resolve.

a) If all of the cells two away are occupied, what does the **Jumper** do? (You don't want an infinite loop.)
b) Do all of the **Blossom**s have the same lifetime? Or are they different?
c) Does a **Jumper** continue to move in one direction if there are no obstacles? Or does it change directions every once in a while?
d) Since **Bug**s can move onto **Flower**s, is it possible for a **Jumper** to move onto a **Blossom**?

Write down on paper what your **Jumper** will do given these situations. Put your name on the paper since Mr Greenstein will use this for grading.

F) Download the **JumperRunner** from the web site to test **Jumper**. The code is shown below.

```
import info.gridworld.grid.Grid;
import info.gridworld.grid.BoundedGrid;
import info.gridworld.grid.Location;
import info.gridworld.actor.ActorWorld;
import info.gridworld.actor.Actor;
import info.gridworld.actor.Bug;
import info.gridworld.actor.Rock;
import info.gridworld.actor.Flower;

public class JumperRunner
{
    public static void main(String[] args)
    {
        BoundedGrid<Actor> mygrid = new BoundedGrid<Actor>(20,20);
        ActorWorld world = new ActorWorld(mygrid);

        world.add(new Location(0,0),new Jumper());
        world.add(new Location(0,1),new Jumper());
```

```
        world.add(new Location(1,0),new Jumper());
        world.add(new Location(1,1),new Rock());
        world.add(new Location(1,2),new Bug());
        world.add(new Location(0,3),new Bug());

        world.add(new Location(2,6),new Jumper());
        world.add(new Location(2,7),new Jumper());
        world.add(new Location(1,7),new Rock());
        world.add(new Location(1,6),new Rock());
        world.add(new Location(0,7),new Rock());

        world.add(new Location(19,1),new Rock());

        world.show();
    }
}
```