Project 3 Hints

Test first for SingleStepPathingStrategy.

Where must the changes happen?

In superclass of Dude and Fairy create an instance variable (object) of type PathingStrategy interface. And then update in the constructor.

1) **Add instance variable:**

```
private PathingStrategy strategy;
public yourSuperClass (…., PathingStrategy pStrategy)
{
  super(….);
  this.strategy = pStrategy;
}
```

2) **modify nextPosition by commenting out the code and calling computePath, which will be implemented in AStarAlgorithm**

```
private Point nextPosition(WorldModel world, Point destPos) {
   List<Point> path = strategy.computePath(pass parameters);
   if (path.size() == 0)
     return this.getPosition();
   else
     return path.get(0);
}
```

One solution is creating an instance variable (object) based on Pathing Strategy in Fairy and Dude. Its type will be interface and it will be new object of class SingleStepPathingStrategy (or any other pathing strategy)

```
private static final PathingStrategy Fairy_PATHING = new SingleStepPathingStrategy();
```

Then update the constructor:

```
super(id, position, images, 0, actionPeriod, animationPeriod, Fairy_PATHING);
```

do the same in Dude super class:

```
private static final PathingStrategy Dude_PATHING = new SingleStepPathingStrategy();
```

Your AStar algorithm can use HashMap, LinkedList, Set, Priority Queue, or any other data structure you prefer to keep track of openList, closeList, and path.

Each item or Node can hold, point (x, y location), g, h, f costs, and prior node.

The following is an example for creating Predicate and BiPredicate which is used in the computePath method.

```
// Creating predicate
  Predicate<Integer> negative = num -> (num < 0);
  // Calling Predicate method
  System.out.println("<0 : " + negative.test(10));

  //Creating BiPredicate
  BiPredicate<Integer, String> cmp =
       (num, str) ->  { if (num == Integer.parseInt(str))
                    return true;
                  return false; };
// Calling BiPredicate method
```

```
System.out.println("cmp(int,str): " + cmp.test(5, "5"));
System.out.println("cmp(int,str): " + cmp.test(5, "2"));
```

Then in computePath method your withinReach is BiPredicate ➔

withinReach.test(current.getPoint(), end)

In superclass of Dude and Fairy

```
Public (or protected) Point nextPosition(WorldModel world, Point destPos){
   List<Point> path = new AStarPathingStrategy().computePath(
                        getPosition(), // start Point
                        destPos, // end Point
                        p -> world.withinBounds(p)&& !world.isOccupied(p), //canPassTrough
                       this::adjacent, //withinReach
                       PathingStrategy.CARDINAL_NEIGHBORS);
```