fish.cpp

```
 1: // Excerpt from fish.cpp
 2:
 3: void Fish::Move(Environment & env)
 4: // precondition:  Fish stored in env at Location()
 5: // postcondition: Fish has moved to a new location in env (if possible)
 6: {
 7:   Position newPos = NextLocation(env);
 8:   if (newPos != myPos) {
 9:     Position oldPos = myPos;
10:     DebugPrint(1, "Fish at " + myPos.ToString() + " moves to " + newPos.ToString
());
11:     myPos = newPos;
12:     env.Update(oldPos, *this); // *this means this fish
13:   } else {
14:     DebugPrint(1, "Fish " + ToString() + " didn't move.");
15:   }
16: }
17:
18:
19: Position Fish::NextLocation(const Environment & env) {
20:   Neighborhood nbrs = EmptyNeighbors(env, myPos);
21:   DebugPrint(3, nbrs.ToString());
22:
23:   if (nbrs.Size() > 0)
24:   {
25:     // there were some empty neighbors, so randomly choose one
26:     return nbrs.Select(randomVals.RandInt(0, nbrs.Size() - 1));
27:   } else {
28:     return myPos;
29:   }
30: }
```

memoryfish.h

```
 1: // memoryfish.h - initial version 04/09/2013
 2:
 3: #include "fish.h"
 4: #include "position.h"
 5:
 6: #ifndef _MEMORY_FISH_H
 7: #define _MEMORY_FISH_H
 8:
 9: class MemoryFish : public Fish {   // Fish is the parent class
10:   public:
11:     MemoryFish(int id, const Position & pos);
12:     virtual void Move(Environment & env);
13:
14:   protected:
15:     Position OldLocation() const;
16:
17:   private:
18:     Position oldPos;
19: };
20:
21: #endif
```

memoryfish.cpp

```
 1: #include "memoryfish.h"
 2:
 3: // constructor
 4:
 5: MemoryFish::MemoryFish(int id, const Position & pos)
 6:   : Fish(id, pos)                    // invoke the parent constructor
 7: {
 8:     oldPos = Location().West();
 9: }
10:
11: // public accessing function3
12:
13: void MemoryFish::Move(Environment & env)
14: {
15:   Position temp = Location();          // record current position
16:   Fish::Move(env);                 // invoke inherited version
17:   oldPos = temp;
18: }
19:
20: // protected accessing function
21: Position MemoryFish::OldLocation() const
22: {
23:   return oldPos;
24: }
```

biasedfish.h

```
 1: // biasedfish.h - initial version 04/09/2013
 2:
 3: #include "memoryfish.h"
 4:
 5: #ifndef _BIASED_FISH_H
 6: #define _BIASED_FISH_H
 7:
 8: class BiasedFish : public MemoryFish  {   // MemoryFish is the parent class
 9:   public:
10:     BiasedFish(int id, const Position & pos);
11:     virtual Position NextLocation(const Environment & env);
12: };
13:
14: #endif
```

biasedfish.cpp

```
 1: #include "biasedfish.h"
 2: #include "environ.h"
 3:
 4: const int FORWARD_BIAS = 2;
 5:
 6: // constructor
 7:
 8: BiasedFish::BiasedFish(int id, const Position & pos)
 9:   : MemoryFish(id, pos)                    // invoke the parent constructor
10: {}
11:
12: // public accessing functions
13:
14: Position BiasedFish::NextLocation(const Environment & env) {
15:   int dr = Location().Row() - OldLocation().Row();
16:   int dc = Location().Col() - OldLocation().Col();
17:
18:   cout << "Fish " << ShowMe() << " had dr=" << dr << " and dc=" << dc << endl;
19:
20:   Position forward(Location().Row() + dr, Location().Col() + dc);
21:   Position reverse(Location().Row() - dr, Location().Col() - dc);
22:   Position left(Location().Row() + dc, Location().Col() - dr);
23:   Position right(Location().Row() - dc, Location().Col() + dr);
24:
25:   // we will draw straws to see which direction wins
26:   int straws = 0;
27:
28:   if (env.IsEmpty(forward))
29:     straws += FORWARD_BIAS;             // more likely to go forward then turn
30:   if (env.IsEmpty(left))
31:     straws += 1;
32:   if (env.IsEmpty(right))
33:     straws += 1;
34:
35:   if (straws == 0) {
36:     // cannot move forward, left, or right; perhaps backward?
37:     if (env.IsEmpty(reverse))
38:       return reverse;
39:     else
40:       return Location();     // no move was possible; return current location
41:   } else {
42:     int choice = randomVals.RandInt(0, straws-1);
43:
44:     if (env.IsEmpty(forward)) {
45:       choice -= FORWARD_BIAS;             // twice as likely as a turn
46:       if (choice < 0)
47:         return forward;
48:     }
49:
50:     if (env.IsEmpty(left)) {
51:       choice -= 1;
52:       if (choice < 0)
53:         return left;
54:     }
55:
56:     return right;
57:   }
58: }
```