Some Object Oriented Design Problems

These are "classic" OOP design problems. Try to come up with your own design. After that you can use google or another web search engine to find programs and solutions to these problems in many different languages. In these specifications a point or cell location in a grid is notated "row@column" as in "5@4".

Eight Queens. In chess a queen can attack along any row, column, or diagonal. Locate 8 queens on an 8 by 8 chess board so that no queen can attack any other queen. There are several correct positions. Frist queen is placed randomly (or with mouse click).

How many knights? How many knights can be placed on a board w/o attacking each other. Frist knight is placed randomly (or with mouse click).

Knights tour? From an arbitrary starting position, move a Knight chess piece around a chess board visiting other squares on the board exactly once. How many squares can be visited?

Bouncing Balls. Consider a bounded rectangle (originPoint, extensionPoint) for example (10@10, 210@210) that has N balls inside it. The balls have random radius R determined uniformly in the range 5 to 10 and move in a random walk. A random walk moves 1 space forward any direction depending on chance. There are 8 directions available. For this problem restrict the walk so that at Step i the next step (Step i+1) can't be to return to the previous step (Step i-1). In addition, a ball can't move if it will collide with another ball or a wall. This design does not concern itself with any graphical representation of the simulation.

Simple Blackjack. Represent a deck of 52 cards (4 suits: spades, diamonds, hearts, clubs; 9 number cards: 2 .. 10; and 4 face cards: jack, queen, king, ace). You must be able to shuffle the deck, split the deck and deal hands of 2 cards to 5 players. Number cards have their value, face cards are worth 10, and aces are worth 11 points. The winning player(s) should be determined. The winner(s) have the the largest sum less than or equal to 21. Extension: players can draw extra cards until they bust (go over 21).
**Conway's Game of Life.** Consider a grid of cells (N by N). The cells can either be alive or dead. On each generation (time interval) determine for each cell its next state. All cells state determination is done "simultaneously" -- that is you determine the next state for each cell without changing any cell's state. After determining the next state for all the cells you then change the state of each cell. This process of determining, and then changing, is called a generation. A cell's state is determined by its neighborhood (the cells in possibly 8 adjacent cells). If the count of alive cells in a cell's neighborhood is 0, 1, or greater than 3 the cell dies (overcrowding). If the count is 2 the cell maintains its current state. If the count is 3 the cell stays alive, or becomes alive (is born). The simulation should start with an initial configuration (assuming a 10 by 10 grid) of alive cells and should run for N generations. Try some of these patterns: