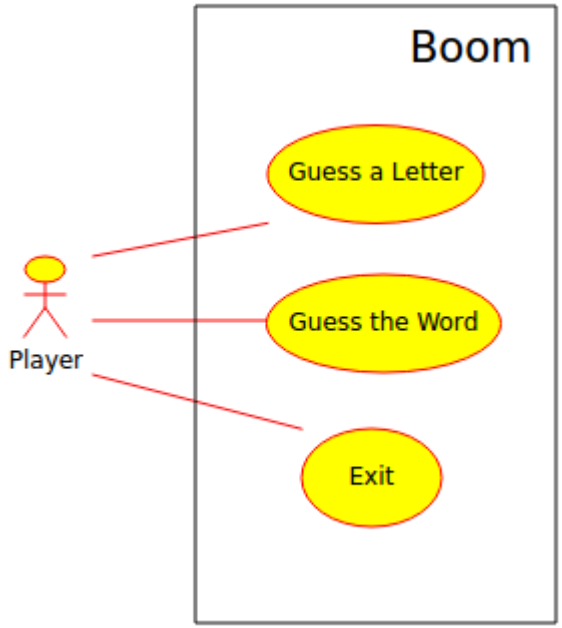
Homework Retrospective
Full Credit UML

Use Case Diagram



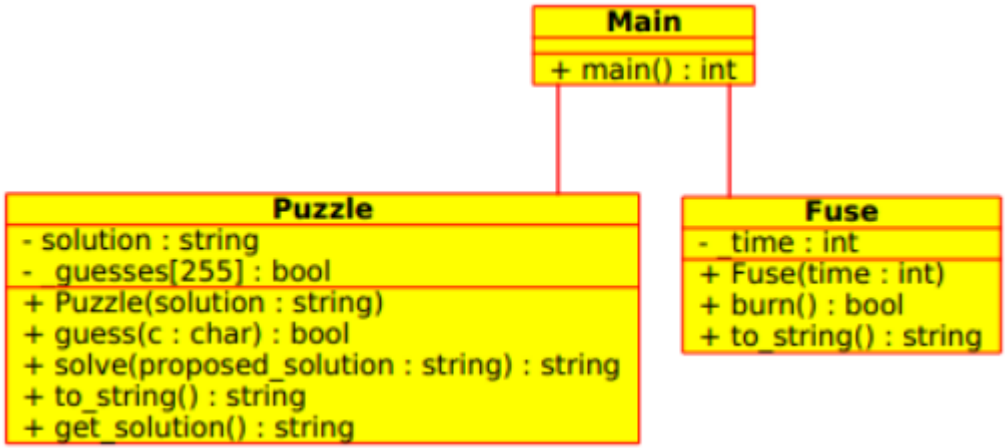
git log

```
ricegf@pluto:~/dev/cpp/201701/P3/xb$ git log
commit 8f5faaa3cd97c01e1ffc6ccacea98b500a0eb648
Author: ricegf <george.rice@uta.edu>
Date:   Mon Jan 30 21:58:01 2017 -0600

    selects random word

commit 8a42f7de528b098778c06eb25a9be512d7bc2061
Author: ricegf <george.rice@uta.edu>
Date:   Mon Jan 30 21:33:11 2017 -0600

    only fizz on a miss

commit ab78e428dd66106b852bb4e31b544bc83e2736c6
Author: ricegf <george.rice@uta.edu>
Date:   Mon Jan 30 21:17:27 2017 -0600

    finalized FC version

commit de40be42f421fc5694e5da49ff49b2f04a3ec908
Author: ricegf <george.rice@uta.edu>
Date:   Mon Jan 30 21:05:08 2017 -0600

    First draft - basic game
ricegf@pluto:~/dev/cpp/201701/P3/xb$
```

Homework Retrospective
Full Credit – Fuse Class

fuse.h

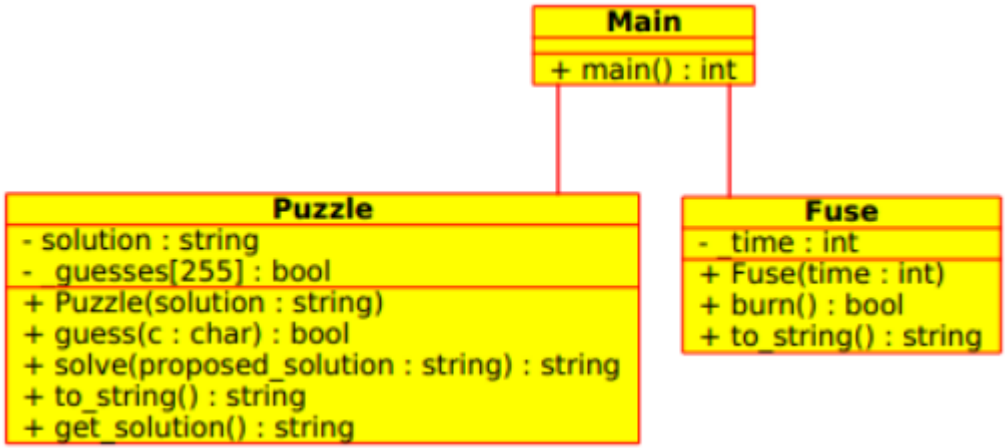
```
#include <string>
using namespace std;

class Fuse {
  public:
    Fuse(int time) : _time{time} { };
    bool burn();  // true if any fuse remain
    string to_string();
  private:
    int _time;
};
```



fuse.cpp

```
#include "fuse.h"

bool Fuse::burn() {
  _time = (_time > 0) ? --_time : _time;
  return _time > 0;
}

string Fuse::to_string() {
  string result = "   ";
  for(int i = 0; i < _time; ++i) result += '_';
  result += "*\n  /\n,+,\n| |\n|_|\n";
  return result;
}
```

Homework Retrospective
Full Credit – Puzzle Class .h



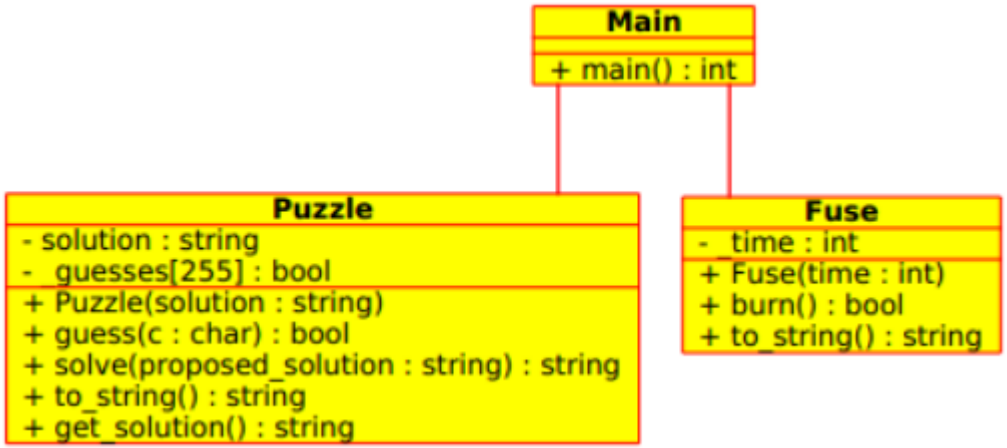
puzzle.h

```
#include <string>
using namespace std;

class Puzzle {
  public:
    Puzzle(string solution) : _solution{solution} {_guesses[' '] = true;}
    bool guess(char c);                    // true if valid guess
    bool solve(string proposed_solution); // true if correctly guessed
    string to_string();
    string get_solution();
  private:
    string _solution;
    bool _guesses[255] = {false};
};
```

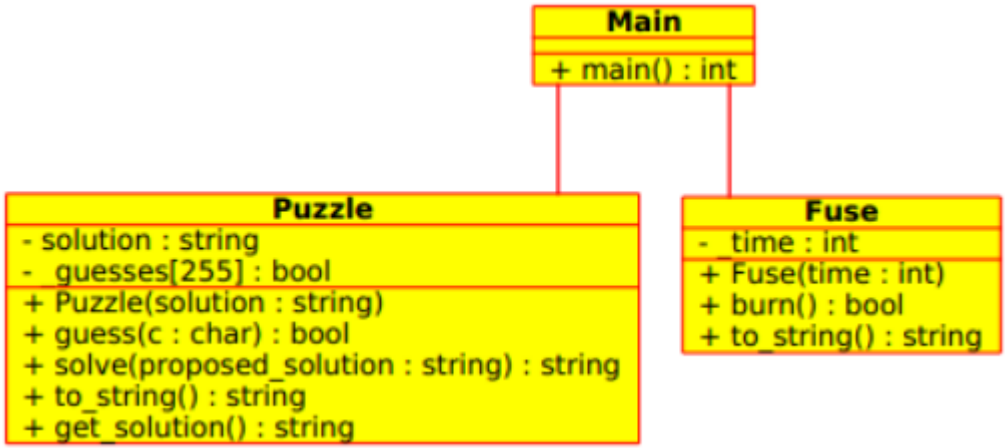
Homework Retrospective
Full Credit – Puzzle Class .cpp

puzzle.cpp

```
#include "puzzle.h"

bool Puzzle::guess(char c) {
  if (c<'a' || c>'z' || _guesses[c])  {
    return false;
  } else {
    _guesses[c] = true;
    return true;
  }
}

bool Puzzle::solve(string proposed_solution) {
  return (proposed_solution == _solution);
}

string Puzzle::to_string() {
  string result = "";
  for (char c: _solution)
    result += _guesses[c] ? c : '_';
  return result;
}

string Puzzle::get_solution() {
  return _solution;
}
```



Homework Retrospective
Full Credit – Main (1 of 2)



main.cpp

```
#include <iostream>
#include "puzzle.h"
#include "fuse.h"

using namespace std;

int main() {
  Puzzle puzzle
      {"a stitch in time saves nine"};
  Fuse fuse{8};
  char guess;
  string proposed_solution;
  bool winner = false;   // true if the player won

  cout << "         ================" << endl
       << "             B O O M !"     << endl
       << "         ================" << endl << endl;
  cout << "Enter lower case letters to guess, " << endl
       << "! to propose a solution," << endl
       << "0 to exit." << endl << endl;
```

Homework Retrospective
Full Credit – Main

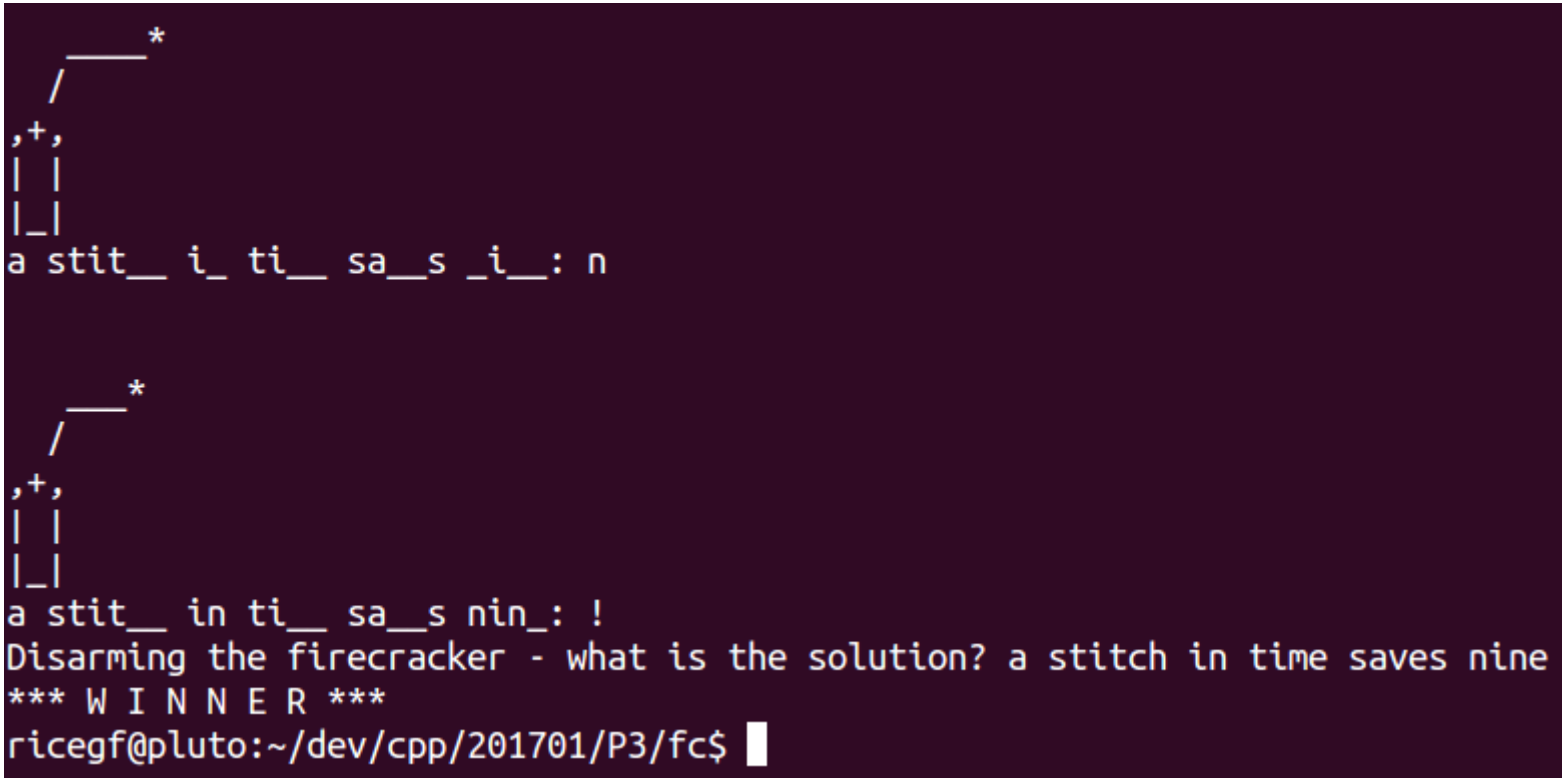
main.cpp

```
  while(true) {
    cout << endl << endl << fuse.to_string() << puzzle.to_string() << ": ";
    cin >> guess;
    if (guess == '0') {
      exit(0);
    } else if (guess == '!') {
      cout << "Disarming the firecracker - what is the solution? ";
      cin.ignore();
      getline(cin, proposed_solution);
      winner = puzzle.solve(proposed_solution);
      break;
    } else {
      if (puzzle.guess(guess)) {
        if (!fuse.burn()) break;
       } else {
        cerr << "Invalid character - try again" << endl;
      }
    }
  }

  if (winner) {
    cout << "*** W I N N E R ***" << endl;
  } else {
    cout << "###### BOOM ######" << endl;
    cout << "The answer was: '" << puzzle.get_solution() << "'" << endl;
  }
}
```

Homework Retrospective
Full Credit – Makefile

```
# Makefile for Boom
CXXFLAGS += --std=c++11

all: main

debug: CXXFLAGS += -g
debug: main

rebuild: clean main

main: main.o puzzle.o fuse.o
	$(CXX) $(CXXFLAGS) main.o puzzle.o fuse.o
main.o: main.cpp puzzle.h fuse.h
	$(CXX) $(CXXFLAGS) -c main.cpp
puzzle.o: puzzle.cpp puzzle.h
	$(CXX) $(CXXFLAGS) -c puzzle.cpp
fuse.o: fuse.cpp fuse.h
	$(CXX) $(CXXFLAGS) -c fuse.cpp
clean:
	-rm -f *.o *~ a.out
```

Homework Retrospective
Full Credit – Game in Progress



Homework Retrospective
Bonus – Puzzle Class

puzzle.h

```
#include <string>
#include <exception>
using namespace std;

class Puzzle {
  public:
    Puzzle(string solution) : _solution{solution} {_guesses[' '] = true;}
    class Bad_char : public exception { };
    bool guess(char c);                          // true if char is in solution
                                                 // throws Bad_char is invalid
    bool solve(string proposed_solution); // true if correctly guessed
    string to_string();
    string get_solution();
  private:
    string _solution;
    bool _guesses[255] = {false};
};
```

puzzle.cpp

```
bool Puzzle::guess(char c) {
  if (c<'a' || c>'z' || _guesses[c]) throw Bad_char{ };
  _guesses[c] = true;
  for (char a : _solution) {
    if (a == c) return true;
  }
  return false;
}
```

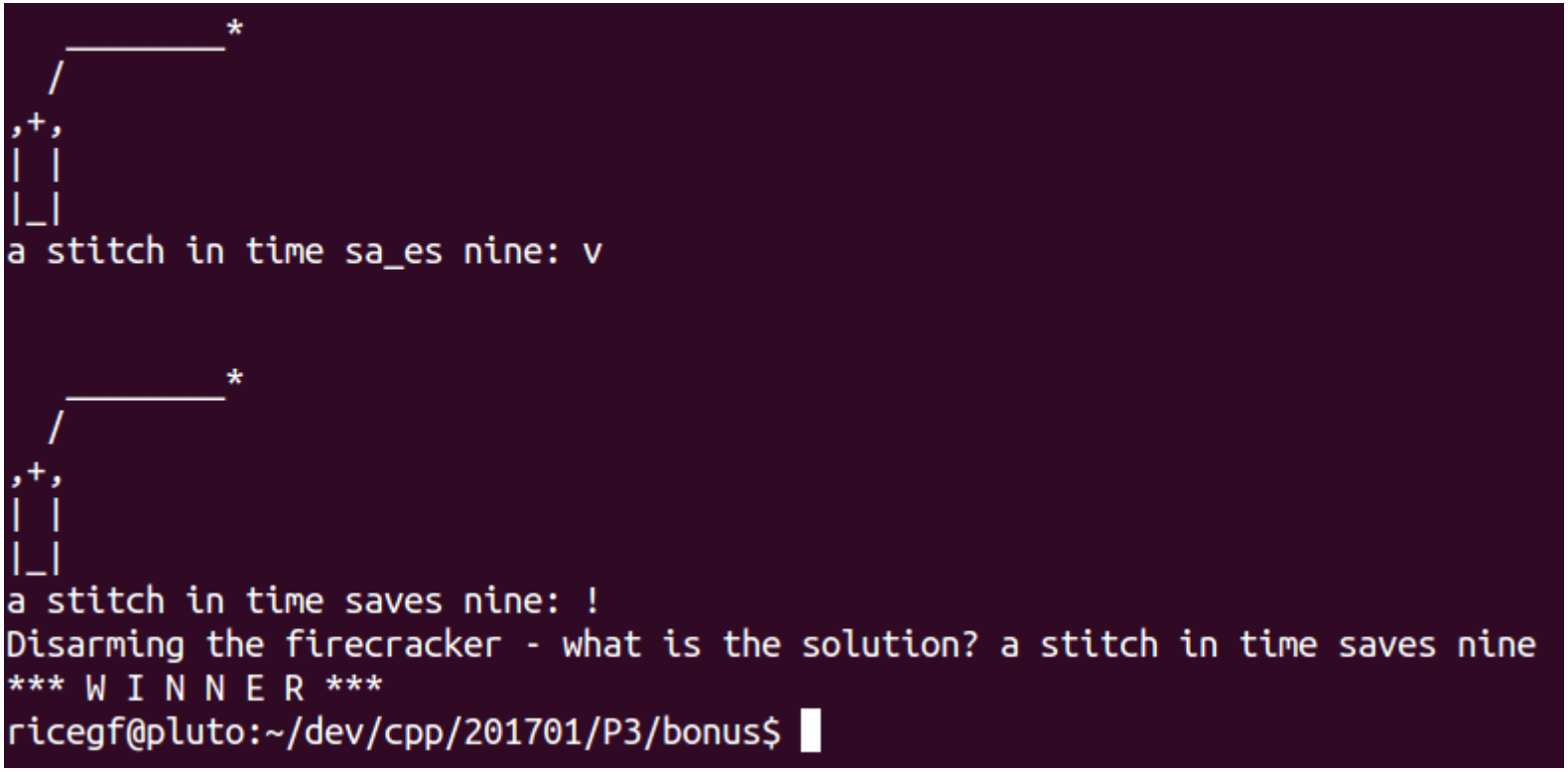
Homework Retrospective
Bonus – Main

main.cpp

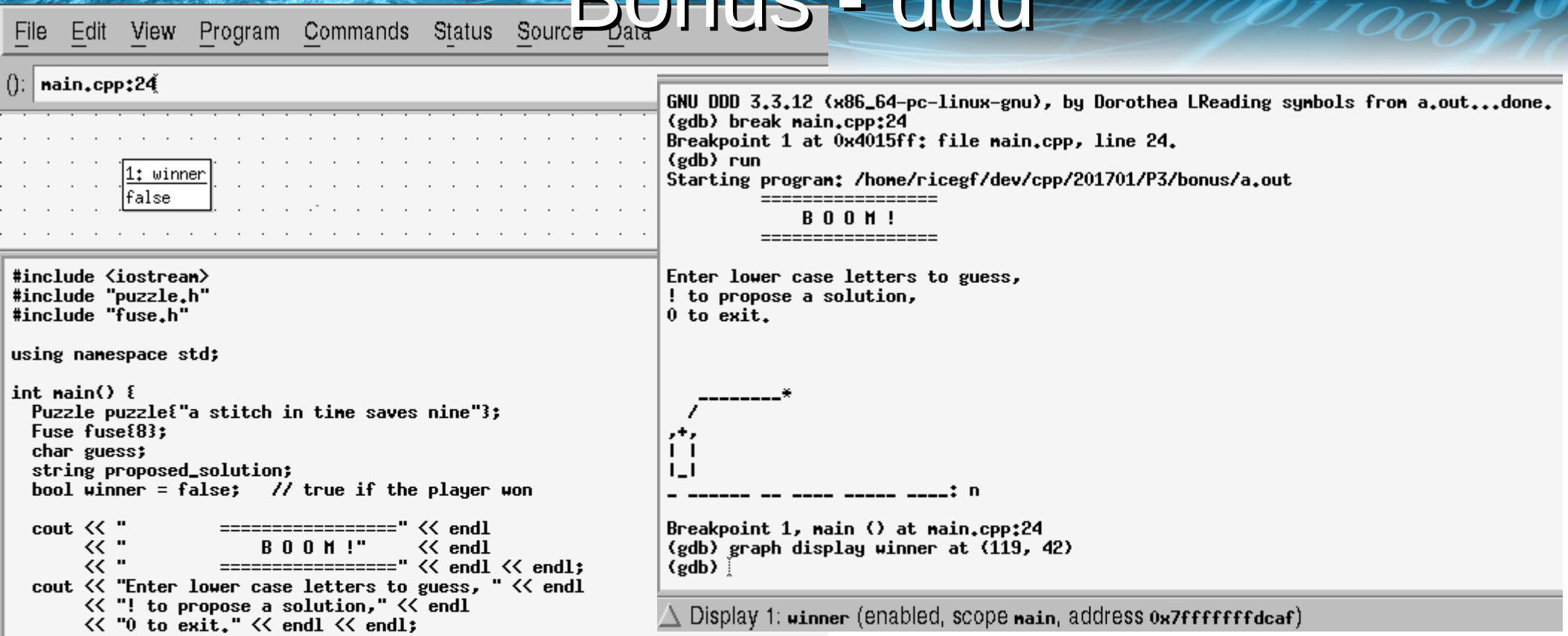
```
while(true) {
  cout << endl << endl << fuse.to_string() << puzzle.to_string() << ": ";
  cin >> guess;
  if (guess == '0') {
    exit(0);
  } else if (guess == '!') {
    cout << "Disarming the firecracker - what is the solution? ";
    cin.ignore();
    getline(cin, proposed_solution);
    winner = puzzle.solve(proposed_solution);
    break;
  } else {
    try {
      if (!puzzle.guess(guess)) {
        if (!fuse.burn()) break;  // fuse burned up – BOOM!
      }
    } catch (Puzzle::Bad_char e) {
      cerr << "Invalid character - try again" << endl;
    }
  }
}
```

No change to the Makefile

Homework Retrospective
Bonus – Game in Progress

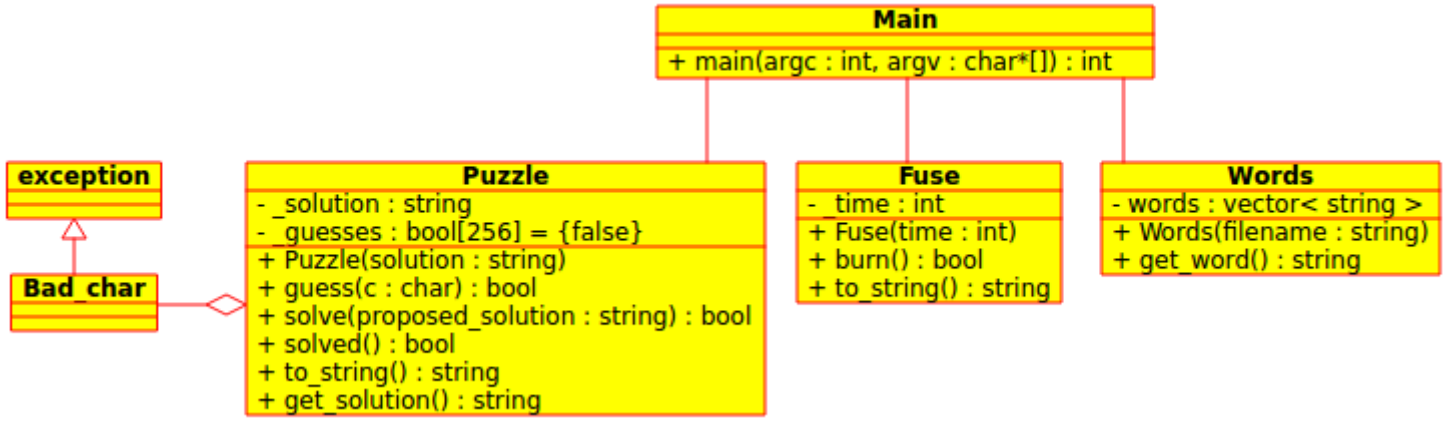


Homework Retrospective
Bonus - ddd




```
#include <iostream>
#include "puzzle.h"
#include "fuse.h"

using namespace std;

int main() {
  Puzzle puzzle{"a stitch in time saves nine"};
  Fuse fuse{8};
  char guess;
  string proposed_solution;
  bool winner = false;   // true if the player won

  cout << "          ==========" << endl
       << "              B O O M !"     << endl
       << "          ==========" << endl << endl;
  cout << "Enter lower case letters to guess, " << endl
       << "! to propose a solution," << endl
       << "0 to exit." << endl << endl;

  while(true) {
    cout << endl << endl << fuse.to_string() << puzzle.to_string() << ": ";
    cin >> guess;
    if (guess == '0') {
      exit(0);
    } else if (guess == '!') {
      cout << "Disarming the firecracker - what is the solution? ";
      cin.ignore();
      getline(cin, proposed_solution);
      winner = puzzle.solve(proposed_solution);
      break;
    } else {
      try {
        if (!puzzle.guess(guess)) {
          if (!fuse.burn()) break;
        }
```

Homework Retrospective
Extreme Bonus – UML



Homework Retrospective
Extreme Bonus – Words

```
#include <string>
#include <vector>

using namespace std;

class Words {
  public:
    Words(string filename);
    string get_word();
  private:
    vector<string> words;
};
```

Words loads a phrase list file into the vector "words" during construction, then returns a random word on request. Puzzle and Fuse are unchanged.

```
#include "words.h"
#include <iostream>
#include <fstream>

string Words::get_word() {
    return words[rand() % words.size()];
}

Words::Words(string filename) {
  srand(time(NULL));
  try {
    string word;
    ifstream ifs;
    ifs.open(filename, ifstream::in);
    while (getline(ifs, word)) {
      words.push_back(word);
    }
    ifs.close();

  } catch(exception e) {
    cerr << "Unable to load wordlist: "
         << filename << endl;
    exit(1);
  }
}
```

Homework Retrospective
Extreme Bonus – Main (1 of 2)

main.cpp

```
int main(int argc, char *argv[]) {

  srand(time(NULL)); // Randomize the random number generator rand()

  // Select a word for the game
  string solution = "a stitch in time saves nine";
  if (argc == 2) {
    Words wordlist{argv[1]};
    solution = wordlist.get_word();
  }
  Puzzle puzzle{solution};
  Fuse fuse{8};
  char guess;
  string proposed_solution;
  bool winner = false;   // true if the player won

  cout << "         =================" << endl
       << "             B O O M !"     << endl
       << "         =================" << endl << endl;
  cout << "Enter lower case letters to guess, " << endl
       << "! to propose a solution," << endl
       << "0 to exit." << endl << endl;
```

This is all that is needed to meet the requirements.
But the program has a serious usability problem –
it doesn't know when the puzzle is solved until you tell it!

Homework Retrospective
Extreme Bonus – Main (2 of 2)

main.cpp

```
while(!winner) {
  cout << endl << endl << fuse.to_string() << puzzle.to_string() << ": ";
  cin >> guess;
  if (guess == '0') {
    exit(0);
  } else if (guess == '!') {
    cout << "Disarming the firecracker - what is the solution? ";
    cin.ignore();
    getline(cin, proposed_solution);
    winner = puzzle.solve(proposed_solution);
    break;
  } else {
    try {
      if (!puzzle.guess(guess)) {
        if (!fuse.burn()) break;
      }
    } catch (Puzzle::Bad_char e) {
      cerr << "Invalid character - try again" << endl;
    }
  }
  winner = puzzle.solved();
}
```

Puzzle

```
bool Puzzle::solved() {
  for (char a : _solution) {
    if (!_guesses[a]) return false;
  }
  return true;
}
```

These three changes also detect when the puzzle is solved by guessing all of the letters.

Homework Retrospective
Extreme Bonus – Makefile

```
# Makefile for Roving Robots
CXXFLAGS += --std=c++11

all: main

debug: CXXFLAGS += -g
debug: main

rebuild: clean main

main: main.o puzzle.o fuse.o words.o
        $(CXX) $(CXXFLAGS) main.o puzzle.o fuse.o words.o
main.o: main.cpp puzzle.h fuse.h words.h
        $(CXX) $(CXXFLAGS) -c main.cpp
puzzle.o: puzzle.cpp puzzle.h
        $(CXX) $(CXXFLAGS) -c puzzle.cpp
fuse.o: fuse.cpp fuse.h
        $(CXX) $(CXXFLAGS) -c fuse.cpp
words.o: words.cpp words.h
        $(CXX) $(CXXFLAGS) -c words.cpp
clean:
        -rm -f *.o *~ a.out
```

Homework Retrospective
Extreme Bonus–Games in Progress