Random random;

It uses the Random class in its shuffle() function. Note that the random object is declared as a private member since it is used only by another member function:

```cpp
void Deck::deal(Hand& hand, unsigned size)
{
    for (int i = 0; i < size; i++)
        hand.cards[i] = cards[top++];
    hand.sort();
}
```

The top member always locates the top of the deck; i.e., the next card to be dealt. So the deal() function copies the top five cards off the deck into the hand's cards array. Then it sorts the hand.

The Deck's constructor initializes all 52 cards in the deck, in the order two of clubs, three of clubs, four of clubs, ..., ace of spades:

```cpp
Deck::Deck()
{
    for (int i = 0; i < 52; i++)
    {
        cards[i].rank_ = Rank(i%13);
        cards[i].suit_ = Suit(i%4);
    }
    top = 0;
}
```

So if hands are dealt without shuffling first, the first hand would be the straight flush of two through six of clubs.

Finally, here is the shuffle() function:

```cpp
void Deck::shuffle()
{
    for (int i = 0; i < 52; i++)
// do 52 random swaps
    {
        int j = random.integer(0, 51);
        Card c = cards[i];
        cards[i] = cards[j];
        cards[j] = c;
    }
    top = 0;
}
```

It swaps the cards in each of the 52 elements with the card in a randomly selected element of the deck's cards array.

12.2 Here are the abstract base classes:

```cpp
const double PI=3.14159265358979;
class Shape
{
    public:
        virtual void print() = 0;
        virtual float area() = 0;
};
class TwoDimensional : public Shape
{
    public:
        virtual float perimeter() = 0;
};
class ThreeDimensional : public Shape
{
    public:
        virtual float volume() = 0;
};
```

Note that the print() function and the area() function prototypes are the same for all classes in this hierarchy, so their interfaces (pure virtual functions) are placed in the Shape base class. But only two-dimensional shapes have perimeters, and only three-dimensional shapes have volumes, so their interfaces are placed in the appropriate second-level ABCs.
Here are two of the seven concrete derived classes:

```cpp
class Circle : public TwoDimensional
{
  public:
    Circle(float r) : radius(r) {}
    void print() { cout << "Shape is a circle.\n"; }
    float perimeter() { return 2*PI*radius; }
    float area() { return PI*radius*radius; }
  private:
    float radius;
};
class Cone : public ThreeDimensional
{
  public:
    Cone(float r, float h) : radius(r), height(h) {}
    void print();
    float area();
    float volume() { return PI*radius*radius*height/3; }
  private:
    float radius, height;
};

void Cone::print()
{
  cout << "Cone: radius = " << radius << ", height = "
       << height << endl;
}
float Cone::area()
{
  float s = sqrt(radius*radius + height*height);
  return PI*radius*(radius + s);
}
```

The other five concrete derived classes are similar.

12.3 Here is the interface for the `Name` class:

```cpp
class Name
{
  friend ostream& operator<<(ostream&, const Name&);
  friend istream& operator>>(istream&, Name&);

  public:
    Name(char*, char*, char*, char*, char*, char*);
    string last() { return last_; }
    string first() { return first_; }
    string middle() { return middle_; }
    string title() { return title_; }
    string suffix() { return suffix_; }
    string nick() { return nick_; }
    void last(string s) { last_ = s; }
    void first(string s) { first_ = s; }
    void middle(string s) { middle_ = s; }
    void title(string s) { title_ = s; }
    void suffix(string s) { suffix_ = s; }
    void nick(string s) { nick_ = s; }
    void dump();

  private:
    string last_, first_, middle_, title_, suffix_, nick_;
};
```