```
public int x() {
    int q = 0;
    int z = 0;
    for (int kk = 0; kk < 10; kk++) {
      if (l[z] == 10)
      {
        q += 10 + (l[z + 1] + l[z + 2]);
        z += 1;
      }
      else if (l[z] + l[z + 1] == 10)
      {
        q += 10 + l[z + 2];
        z += 2;
      } else {
        q += l[z] + l[z + 1];
        z += 2;
      }
    }
    return q;
  }
```

Here is the code the way it should be written. This snippet is actually less complete than the one above. Yet you can infer immediately what it is trying to do, and you could very likely write the missing functions based on that inferred meaning. The magic numbers are no longer magic, and the structure of the algorithm is compellingly descriptive.

```
public int score() {
  int score = 0;
  int frame = 0;
  for (int frameNumber = 0; frameNumber < 10; frameNumber++) {
    if (isStrike(frame)) {
      score += 10 + nextTwoBallsForStrike(frame);
      frame += 1;
    } else if (isSpare(frame)) {
      score += 10 + nextBallForSpare(frame);
      frame += 2;
    } else {
      score += twoBallsInFrame(frame);
      frame += 2;
    }
  }
  return score;
}
```

The power of carefully chosen names is that they overload the structure of the code with description. That overloading sets the readers' expectations about what the other functions in the module do. You can infer the implementation of `isStrike()` by looking at the code above. When you read the `isStrike` method, it will be "pretty much what you expected."[13]

```
private boolean isStrike(int frame) {
  return rolls[frame] == 10;
}
```