

```

1 package club.westcs.OOPNotes;
2
3 import java.util.Random;
4 import java.util.Scanner;
5
6 public class Viking {
7
8     //Attributes
9     /*
10      * the parts of an object
11      * declare the parts but do not assign them values example   private String name;
12      * set to private access modifier
13      *     private
14      *         an access modifier that creates a variable available only to this class
15      */
16     private String name;
17     private boolean alive;
18     private int weapons;
19     private Random rand;
20     private Scanner scan;
21
22     //Constructor
23     /*
24      * gives values to all attributes for each instance of the object
25      * (Puts the object together)
26      * runs automatically when you make a new instance of an object
27      * Always set to public
28      * does not have a return type (no void String....)
29      * Always named and spelled just like the class
30      */
31     public Viking() {
32         rand = new Random();
33         scan = new Scanner(System.in);
34         alive = true;
35         weapons = rand.nextInt(4) + 2; // randomly 2 - 5 weapons
36         name = setName();
37         System.out.println(this.toString());
38     }
39
40     // Viking v = new Viking();

```

```

39
40 public Viking(String name) { //overloaded constructor (if a new instance has a String it will call this one)
41     rand = new Random();
42     scan = new Scanner(System.in);
43     alive = true;
44     weapons = rand.nextInt(4) + 2; // randomly 2 - 5 weapons
45     this.name = name;
46     System.out.println(this.toString());
47 }
48 //Viking v = new Viking("Bob");
49 //Methods
50 /*
51  * The stuff an object can do.
52  * Look like functions but belong to objects.
53  * Can be public or private
54  * Used to get and set values of attributes.
55  */
56 public String setName() {
57     System.out.println("What is this Viking's name?");
58     return scan.nextLine();
59 }
60
61 public String toString() {
62     return this.name + " has " + this.weapons + " weapons.";
63 }
64
65 public boolean isAlive() {
66     return this.alive;
67 }
68
69 public void setAlive() {
70     if(this.alive && this.weapons <= 0) {
71         System.out.println(this.name + " has fallen and gone to Valhalla.");
72         this.alive = false;
73     }
74 }
75
76 public void loseAWeapon() {
77     System.out.println("Grrrrrrhrrrrh");
78     System.out.println(this.name + " has lost a weapon.");
79     this.weapons--;
80     setAlive();
81 }
82
83 public void attack(Viking other) {
84     System.out.println(this.name + " has attacked " + other.getName() + ".");
85     if(other.isAlive() && rand.nextBoolean()) {
86         other.loseAWeapon();
87     }
88 }
89
90 public String getName() {
91     return this.name;
92 }
93 }
94

```