Encapsulation and Information Hiding

- Users of a class and its objects are called its clients. The class that provides the objects is called the server.
- Author of a class is called its implementor.
- Information hiding principle: Implementors hide implementation details inside a class definition so clients can’t see them. In particular, they make all fields private.
- public methods of a class provide services to clients. The interface of a class is the contract in which the implementor commits to deliver certain services to the clients.
- Implementors are free to change the implementation of a class provided the interface doesn’t change, i.e., provided the clients can’t see any change in services.

Client-Server Model

- Structure program as a collection of classes.
  - Some classes have general utility.
  - Other classes are specific to the application at hand.
- Design of adventure game and implementation
  - Class Room
    - collection of rooms numbered starting at 1
    - rooms connected to one another by tunnels reached via doors
    - no room has more than 3 doors
  - Class Game
    - client of class Room
    - processes input to build cave system
    - processes moves and keeps track of player position and monster position
    - creates output
    - stops game when player finds exit or when player and monster are in same room

Pattern for Searching

// Start at the first place to look
r = the first place to look ;

while ( there are still more places to look and r is not what we are looking for )
  r = the next place to look ;

// Now r is either what we were looking for or
// there is an indication that there were no more places to look.
// Question: is the order of the conditions in the loop guard important?

Output Statement

- Output statement: System.out.println( expression )
- If the type of expression is String (i.e., text), then the value of expression is sent to output.
- Otherwise, the value of expression is first converted to a string and then sent to output.
- Example:
  System.out.println( "even" ); // output string: even
  System.out.println( 70 ); // output string: 70
Output of Objects

- An object o is “converted to a string” by method o.toString()
- Every object has a default definition of toString(). Suppose rA is a reference to an object:
  System.out.println( rA );  //output: something cryptic
- A class can redefine toString:
  
    public String toString(){
      return expression;
    }

Reference Equality

- Two references to objects are equal if they refer to one and the same object.
- E.g., suppose there is a class Foo defined, what would be the output of the following code segment?
  Foo p1 = new Foo();  Foo p2 = new Foo();
  if ( p1 == p2 )
    System.out.println(“same object”);
  else
    System.out.println(“different objects”);

Examples of tests in test harness

public static void main(String args [])
{
  Room rA = new Room();
  Room rB = new Room();
  Room rC = new Room();

  // Validate toString.
  System.out.print(“Here is room 1”);
  System.out.println( rA );
  System.out.print(“Here is room 4”);
  System.out.println( new Room() );

  // Validate connect and farRoom.
  Room.connect(rA, rB);
  if ( rA.farRoom(1)!= rB ) System.out.println(“Connect failure 1”);
  if ( rB.farRoom(1)!= rA ) System.out.println(“Connect failure 2”);
  if ( rA.farRoom(2)!= null ) System.out.println(“Connect failure 3”);
  if ( rB.farRoom(2)!= null ) System.out.println(“Connect failure 4”);

  Room.connect(rA, rC);
  if ( rA.farRoom(1)!= rB ) System.out.println(“Connect failure 5”);
  if ( rA.farRoom(2)!= rC ) System.out.println(“Connect failure 6”);
  if ( rC.farRoom(2)!= rA ) System.out.println(“Connect failure 7”);

  . . .
}

Beginning of the application class

- read input on number of rooms and create all rooms
- read input on cave configuration and make all connections
- start the game!