Command Design Pattern

Rick Mercer
The Command Pattern encapsulates a request as an object, thereby letting you parameterize other objects with different requests, queue commands, send commands over the internet like in a networked game, and support undo and redo….

http://en.wikipedia.org/wiki/Command_pattern
Example for Heads First

Note: RemoteLoader loads commands into slots of the remote control
Command Pattern

- One object can send messages to other objects without knowing anything about the actual operation or the type of object.
- Polymorphism lets us encapsulate a request for services as an object.
  - Establish a method signature name as an interface.
  - Vary the algorithms in the called methods.
Java Examples

- Sun used the Command pattern to improve the event model in Java 1.1
- one example method signature:
  ```java
  public void actionPerformed(ActionEvent e)
  ```
  - JButtons and JTextFields send `actionPerformed` messages to "command" objects (the listeners) without knowing what will happen
- Event generators — buttons, text fields, mouse — have listener objects (actually a List of listener objects)
Uses

- The Command object can also be used when you need to tell the program to execute the command later.
  - In such cases, you are saving commands as objects to be executed later.

- You could also sending command objects over the network (in new project) or save them in a collection class such as a Stack for undo/redo operations.
Example we saw before was Command

- Make 3 command classes
- Log instances by writing the objects to a file
  - See SaveWorkCommands.java in the Eclipse project CommandPattern (command.zip)
  - Like RemoteLoader in HFSP or Client in gen. form
- Read the objects later and execute them
  - See ExecuteSavedCommands.java in the Eclipse project CommandPattern (command.zip)
  - Like Light on HFSP or Receiver in gen. form
import java.io.Serializable;

/** Command design pattern - Decoupling producer from consumer. */

public interface WorkCommand {
    void execute();
}

class DomesticEngineer implements WorkCommand, Serializable {
    public void execute() {
        System.out.println("Take out the trash.");
    }
}

class Politician implements WorkCommand, Serializable {
    public void execute() {
        System.out.println("Take money from the rich, take votes from the poor.");
    }
}

class Programmer implements WorkCommand, Serializable {
    public void execute() {
        System.out.println("Sell the bugs, charge extra for the fixes.");
    }
}

See code demos page for CommandPattern.zip
Summary

- The Command design pattern encapsulates the concept of a command into an object.
- A command object could be sent across a network to be executed elsewhere or it could be saved as a log of operations.
References


- **[Sierra and Bates]**, Heads First Design Patterns