This code demonstrates the **Interpreter design pattern**, where you define a representation for a language's grammar and use that representation to interpret specific commands. The pattern is often used to evaluate sentences or expressions in a specific language.

**Class Structure:**

1. **Expression Interface:**
   * This interface defines a common method interpret() for all expressions. Each type of command or operation will implement this interface, providing its own interpretation logic.
2. **Concrete Expressions:**
   * These are implementations of the Expression interface, each representing a different command:
   * **PrintExpression:**
     + This class is responsible for handling "Print" commands. It takes a string message as input and prints it.
     + Constructor PrintExpression(String message) initializes the message to be printed.
     + The interpret() method prints the message prefixed with "Print:".
   * **AddExpression:**
     + This class handles "Add" commands. It takes two integers (a and b) as input, adds them, and prints the result.
     + The interpret() method prints the sum of the two integers.
   * **ExitExpression:**
     + This class handles the "Exit" command.
     + The interpret() method prints "Exiting..." and terminates the program using System.exit(0).
3. **CommandLineInterpreter (Interpreter):**
   * This class serves as the interpreter for processing input commands. It evaluates the command and returns the corresponding Expression object based on the command type.
   * It checks the command string and:
     + For commands starting with "Print", it creates and returns a PrintExpression object.
     + For commands starting with "Add", it parses the integers from the command string, creates, and returns an AddExpression object.
     + For "Exit", it returns an ExitExpression object.
     + If the command doesn't match any known pattern, it returns null.