Name: **Shruti Patil**

Student ID : 010827622

Design Pattern: Command

Identifying areas to use Command Pattern in Team Project- Game.

Command Patterns is used for various modules in game.

Configuring Input: A chunk of code that reads in raw user input — button presses, keyboard events, mouse clicks, whatever. It takes each input and translates it to a meaningful action in the game.

Such as

- startGame(),
- exitGame()

Command Patterns also will be used for selecting the difficulty levels:

- easy(),
- medium()
- hard()

Dead simple implementataion looks like this:

```
void handleInput()
{
  if (isPressed(BUTTON_A)) startGame();
  else if (isPressed(BUTTON_B))ExitGame();

  else if (isPressed(BUTTON_X)) easy();
  else if (isPressed(BUTTON_Y)) medium();
  else if (isPressed(BUTTON_Z)) hard();
  }
```

we need an *object* that we can use to represent a game action.

We define a base class that represents a triggerable game command:

```
class Command
{
public:
  virtual ~Command() {}
  virtual void execute() = 0;
};
```

Then we create subclasses for each of the different game actions:

```
class startCommand extends Command
{
public:
  virtual void execute() { startGame(); }
};

class ExitCommand extends Command
{
public:
  virtual void execute() { exitGame(); }
};
```

In our input handler, we store a instance to a command for each button:

```
class InputHandler
{
public:
  void handleInput();



private:
  Command buttonX_ = new Command();
  Command buttonY_ = new Command();
  Command buttonA_ = new Command();
  Command buttonB_ = new Command();
  Command buttonZ_= new Command();

};
```

Now the input handling just delegates to those:

```
void handleInput()
{
  if (isPressed(BUTTON_X)) button_.execute();
  else if (isPressed(BUTTON_Y)) buttonY_.execute();
  else if (isPressed(BUTTON_A)) buttonA_.execute();
  else if (isPressed(BUTTON_B)) buttonB_.execute();
}
```

Thus, we have seen the dead simple code as base for Command Pattern.