Java Programming II

Lab6

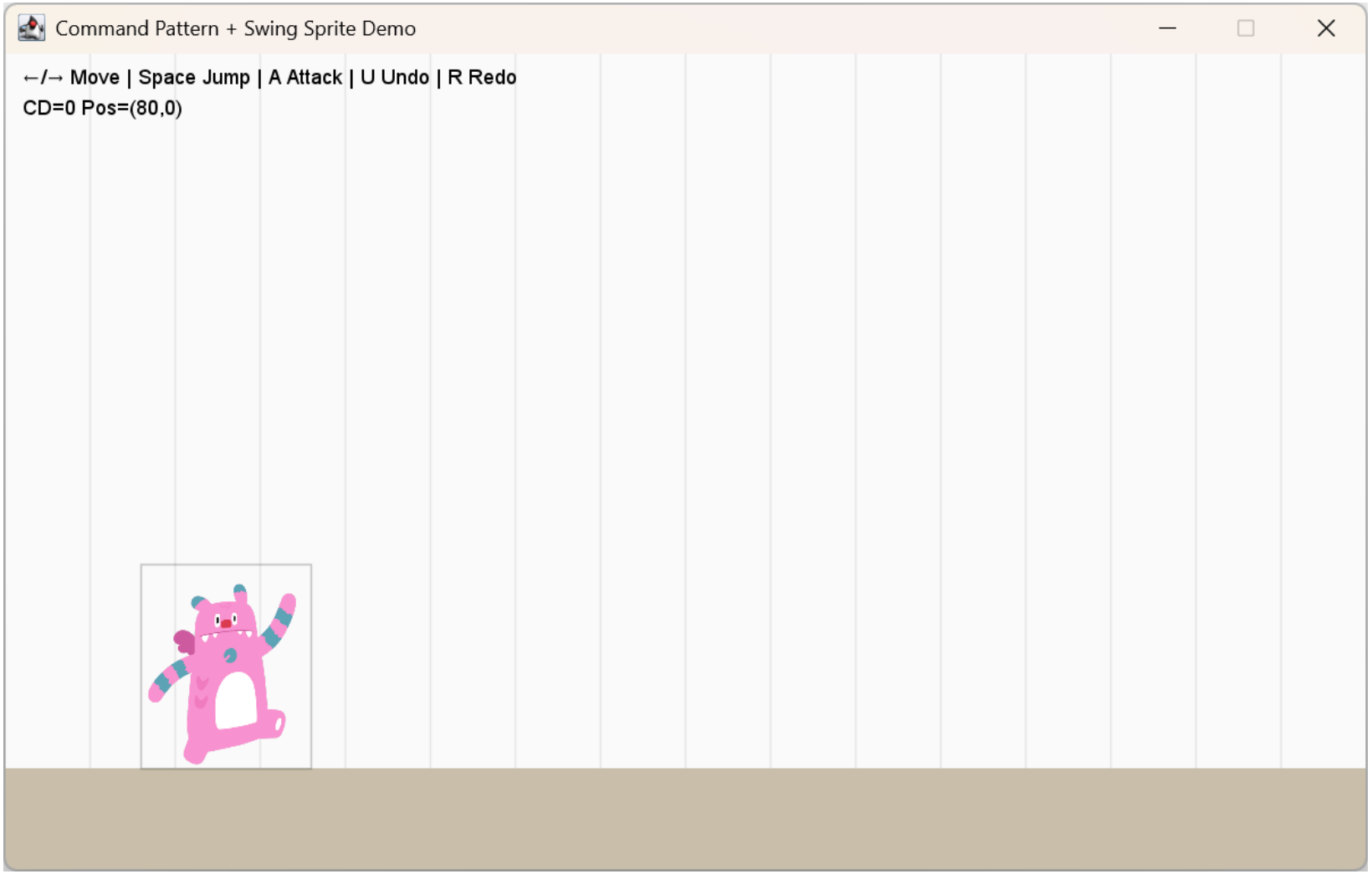
514770-1
Fall 2025
11/5/2025

Kyoung Shin Park
Computer Engineering
Dankook University

Lab6

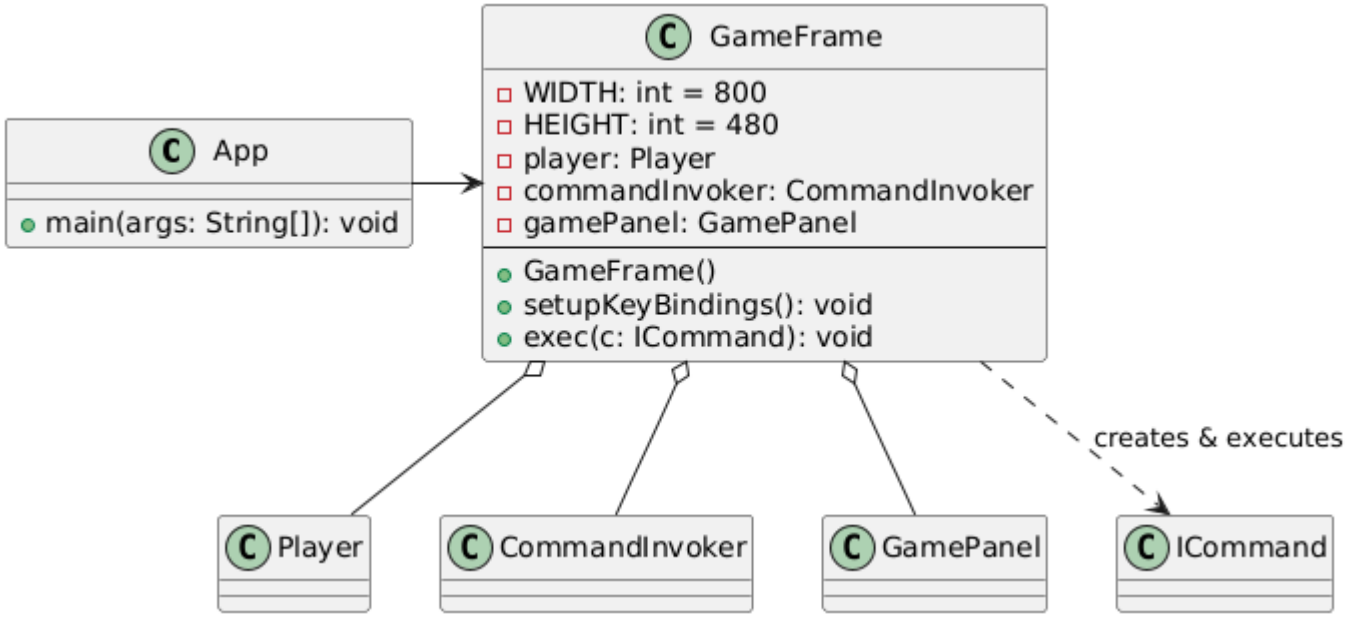
- Practice to write an **GamePlayer control program** using **Command pattern.**
  - **ICommand** interface has execute() and undo() and default String name() which returns getClass().getSimpleName().
  - **Move, Jump, Attack, and MacroCommand** implements **ICommand** to make the player to move, jump, attack.
  - Player class move, jump, attack, and tickPhysics.
  - **CommandInvoker** is the invoker class, and uses Deque<ICommand> undoStack and Deque<ICommand> redoStack.
  - GamePanel class extends JPanel and override paintComponents(Graphics g).
  - GameFrame class extends JFrame loads a sprite image from a file (e.g. haechi.png) and uses Player, GamePanel, CommandInvoker to set commands.
  - App calls GameFrame (extends JFrame) to create a Swing app.

Lab6


Command Pattern + Swing Sprite Demo
←/→ Move | Space Jump | A Attack | U Undo | R Redo
CD=0 Pos=(80,0)

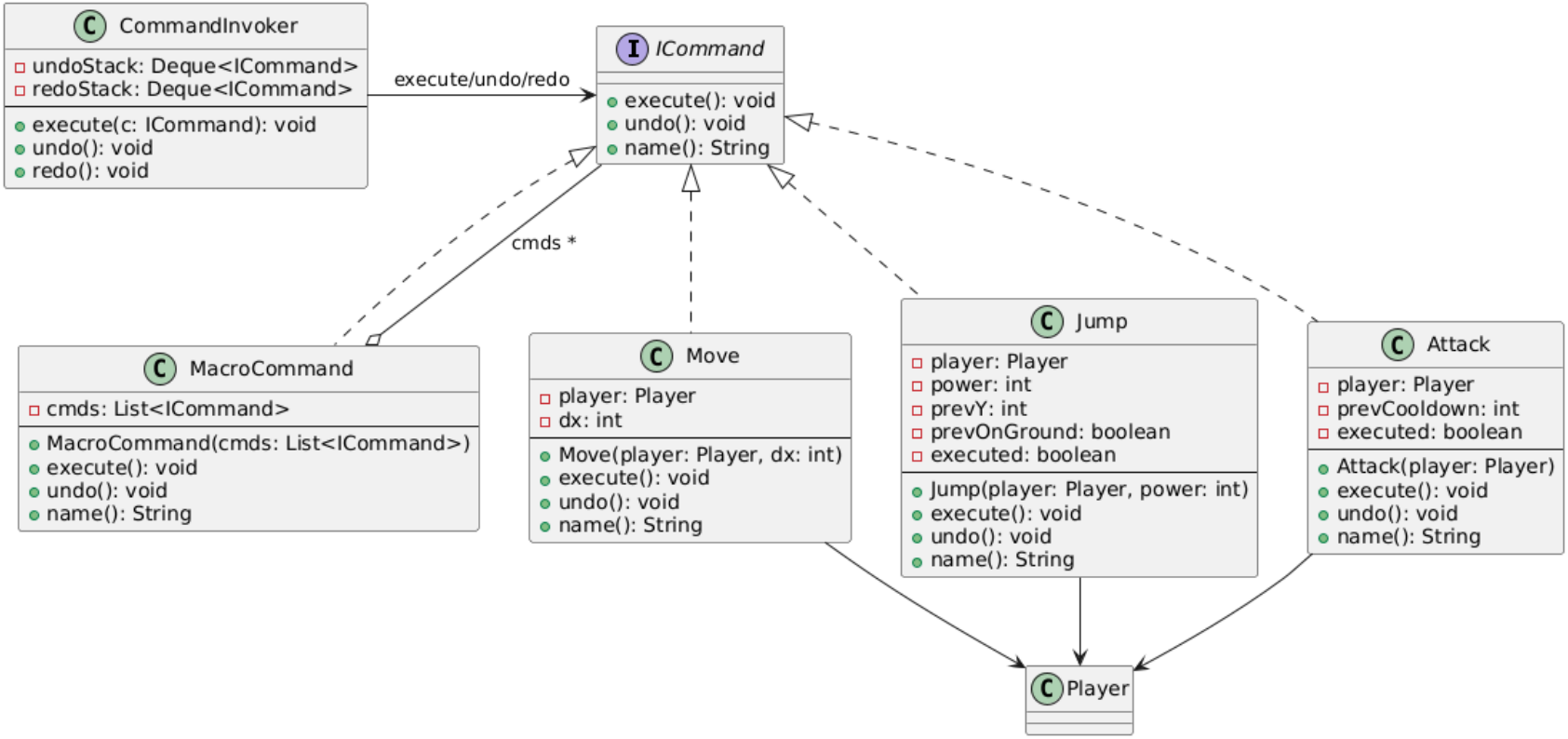
Lab6

- The App client call GameFrame which uses **CommandInvoker** to **set** command and then, **execute** the command.



Lab6

- **CommandInvoker execute/undo/redo** the command.
  - Execute(ICommand c): **c.execute()** -> push undo & clear redo
  - undo(): pop undoStack -> **c.undo()**; push to redo
  - redo(): pop redoStack -> **c.execute()**; push to undo



Lab6

```
public class Move extends ICommand {
  private final Player player;
  private final int dx;
…
  @Override
  public void execute(){
    player.move(dx);
  }
  @Override
  public void undo(){
    player.move(-dx);
  }
  @Override
  public String name(){
    return "Move(" + dx + ")";
  }
}
```

Lab6

```
public class Player {
    Color baseColor = new Color(60, 140, 255);
    int spriteW = 50;
    int spriteH = 60;
    int x = 80;      // X position on screen (starts from left)
    int vx = 0;      // X velocity (instant movement)

    int y = 0;       // Y position (height above ground, 0 is ground)
    boolean onGround = true; // True if player is on the ground

    int attackCooldown = 0; // attack cooldown (frames)
    Color auraColor = null;  // attack aura color
    int auraTicks = 0; // aura remaining duration (frames)

    String bubbleText = null; // bubbleText near Player
    int bubbleTicks = 0; // remaining display time (frames)

    public void move(int dx) {         x += dx;     } ……
}
```

Lab6

```
public class GamePanel extends JPanel {
    private static final double SCALE = 2.0;
    private final Player player;
    private final int groundY;
    private final Image sprite;

    public GamePanel(Player p, int width, int height, Image sprite) {
        this.player = p;
        this.sprite = sprite;
        setPreferredSize(new Dimension(width, height));
        setBackground(new Color(250, 250, 250));
        groundY = height - 60;
        setDoubleBuffered(true);
        setFocusable(true);
    }
…
```

Lab6

```
    @Override
    protected void paintComponent(Graphics g) {
        // draw background/ground
        // draw grid
        // calculate player's location
        // draw attack aura
        // draw player sprite image
        // draw BubbleText near Player to show Undo Jump
        // draw HUD "←/→ Move | Space Jump | A Attack | U Undo | R Redo"
    }
}
```

Lab6

```
public class GameFrame extends JFrame {
    private static final int WIDTH = 800;
    private static final int HEIGHT = 480;

    private final Player player = new Player();
    private final CommandInvoker invoker = new CommandInvoker();
    private final GamePanel gamePanel;

    public GameFrame() {
        super("Command Pattern + Swing Sprite Demo");
        Image sprite = new ImageIcon("haechi.png").getImage();
        this.gamePanel = new GamePanel(player, WIDTH, HEIGHT, sprite);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setContentPane(gamePanel); // attach GamePanel
        pack();
        setLocationRelativeTo(null);
        setResizable(false); // fixed size
.. Continue..
```

Lab6

```
        // set Game Loop Timer (60 FPS)
        new Timer(1000/60, _ -> {
            player.tickPhysics();
            gamePanel.repaint();
        }).start();

        // set KeyBindings
        setupKeyBindings();

        setVisible(true);
    }

    // Repaint immediately after executing the command
    private void exec(ICommand c) {
        invoker.execute(c);
        gamePanel.repaint();
    }
```

Lab6

```
    // Set key bindings for game actions
    private void setupKeyBindings() {
        final InputMap im =
gamePanel.getInputMap(JComponent.WHEN_IN_FOCUSED_WINDOW);
        final ActionMap am = gamePanel.getActionMap();
        // Move Command
        bind(im, am, "LEFT",  () -> exec(new Move(player, -10)));
        bind(im, am, "RIGHT", () -> exec(new Move(player, +10)));
        // Jump Command
        bind(im, am, "SPACE", () -> exec(new Jump(player, 100)));
        // Attack Command
        bind(im, am, "A",     () -> exec(new Attack(player)));
        // Undo Command
        bind(im, am, "U",     () -> { invoker.undo();
gamePanel.repaint(); });
        // Redo Command
        bind(im, am, "R",     () -> { invoker.redo();
gamePanel.repaint(); });
    }
```

Lab6

```
    // KeyStroke binding helper
    private void bind(InputMap im, ActionMap am, String keyStroke,
Runnable r) {
        // Register in ActionMap with KeyStroke name
        im.put(KeyStroke.getKeyStroke(keyStroke), keyStroke);

        // Define the actual action for a KeyStroke
        am.put(keyStroke, new AbstractAction() {
            @Override
            public void actionPerformed(ActionEvent e) {
                r.run();
            }
        });
    }
}
```

Submit to e-learning

- Add your code (e.g., additional method, class, routine, etc) in the Lab6 assignment.
- Submit the Lab6 assignment (JAVA25-2-Lab6-ID-name.zip including the report) to e-learning (due by 11/11).