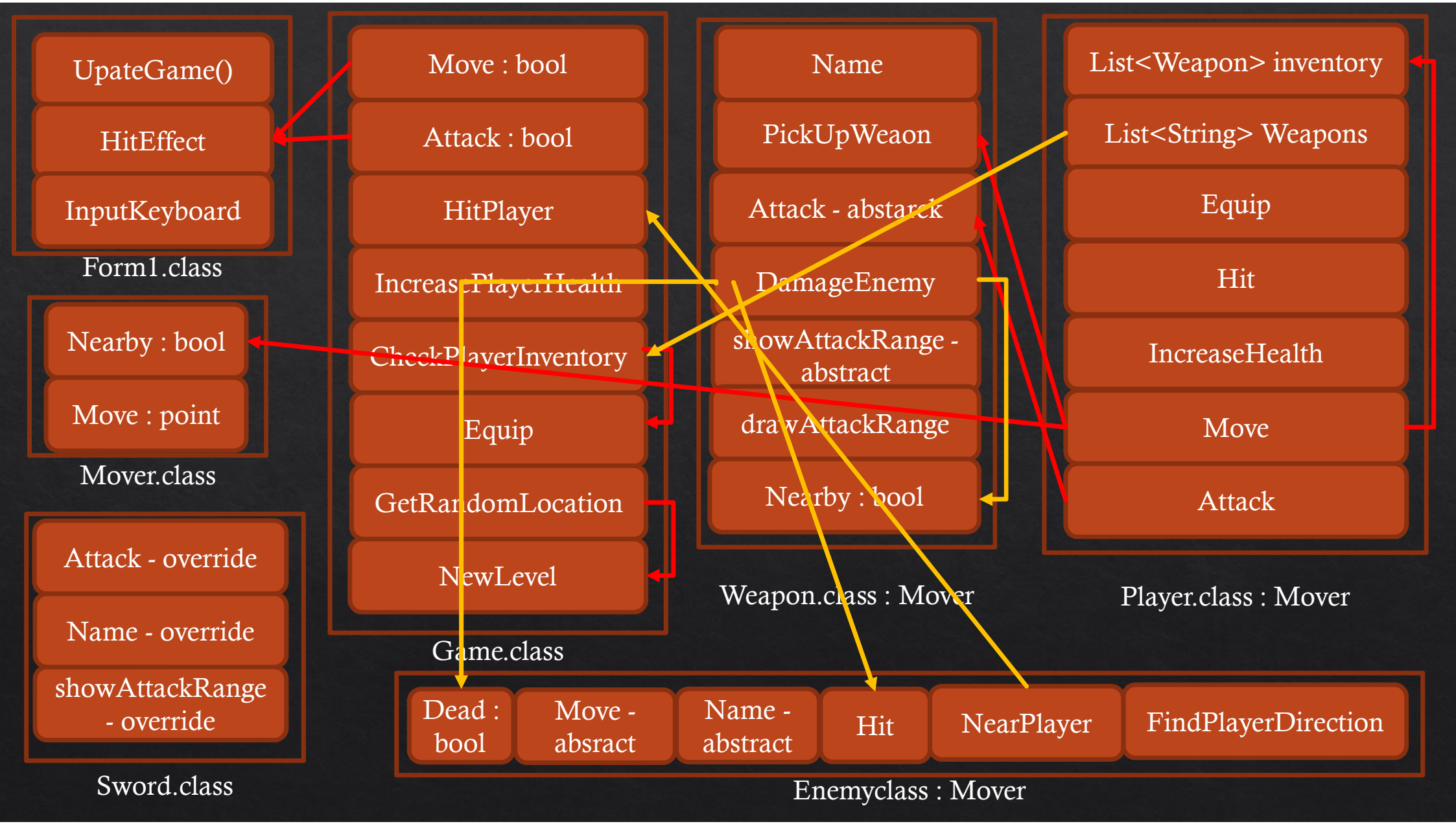
# The Quest

박지용


UpateGame()
HitEffect
InputKeyboard
Form1.class
Nearby : bool
Move : point
Mover.class
Attack - override
Name - override
showAttackRange - override
Sword.class
Move : bool
Attack : bool
HitPlayer
IncreasePlayerHealth
CheckPlayerInventory
Equip
GetRandomLocation
NewLevel
Game.class
Name
PickUpWeaon
Attack - abstarck
DamageEnemy
showAttackRange - abstract
drawAttackRange
Nearby : bool
Weapon.class : Mover
List<Weapon> inventory
List<String> Weapons
Equip
Hit
IncreaseHealth
Move
Attack
Player.class : Mover
Dead : bool
Move - absract
Name - abstract
Hit
NearPlayer
FindPlayerDirection
Enemyclass : Mover

# Form1.class

```
foreach (Enemy enemy in game.Enemies)
{
    switch (enemy.Name)
    {
        case "Bat":
            Bat.Location = enemy.Location;
            batHitPoints.Text = enemy.HitPoints.ToString();
            if (enemy.HitPoints > 0)
            {
                showBat = true;
                enemiesShown++;
            }
            break;
        case "Ghost":
            Ghost.Location = enemy.Location;
            ghostHitPoints.Text = enemy.HitPoints.ToString();
            if (enemy.HitPoints > 0)
            {
                showGhost = true;
                enemiesShown++;
            }
            break;
        case "Ghoul":
            Ghoul.Location = enemy.Location;
            ghoulHitPoints.Text = enemy.HitPoints.ToString();
            if (enemy.HitPoints > 0)
            {
                showGhoul = true;
                enemiesShown++;
            }
            break;
        default: break;
    }
}
// 조건문 ? 참 : 거짓
Bat.Visible = showBat ? true : false;
Ghost.Visible = showGhost ? true : false;
Ghoul.Visible = showGhoul ? true : false;
}
```

```
public void EffectThread()
{
    Thread effect = new Thread(new ThreadStart(HitEffect));
    effect.Start();
}

public void HitEffect() // 반투명 빨강 이펙트
{
    Form bg = this;
    bg.BackgroundImage = Properties.Resources.hitEffect;
    System.Threading.Thread.Sleep(150);
    bg.BackgroundImage = Properties.Resources.dungeon600x400;
}
```

```
// 키보드 입력 처리
private void InputKeyboard(object sender, KeyEventArgs e)
{
    if (e.KeyCode == Keys.D1 || e.KeyCode == Keys.NumPad1)
    {
        if (previous != null) previous.BorderStyle = BorderStyle.None;

        game.Equip("Sword");
        InvenSword.BorderStyle = BorderStyle.FixedSingle;

        previous = InvenSword;
    }
```

Weapon.class

```
public abstract void Attack(Direction dir, Random random, Form1 form);

public abstract void showAttackRange(Direction dir, Form1 form);
public void drawAttackRange(Direction dir, int attackRange, Form1 form)
{
    Color hiteffect = Color.FromArgb(0x30, 0x00, 0x00, 0xff);
    Graphics graphics = form.CreateGraphics();
    Pen pen = new Pen(hiteffect, 1.0f);
    Rectangle rec;

    int size = 30;

    switch (dir)
    {
        case Direction.up:
        case Direction.Up:
            rec = new Rectangle(game.PlayerLocation.X, game.PlayerLocation.Y - attackRange + size, size, attackRange);
            graphics.DrawRectangle(pen, rec);
            break;
        case Direction.right:
        case Direction.Right:
            rec = new Rectangle(game.PlayerLocation.X, game.PlayerLocation.Y, attackRange, size);
            graphics.DrawRectangle(pen, rec);
            break;
        case Direction.Down:
            rec = new Rectangle(game.PlayerLocation.X, game.PlayerLocation.Y, size, attackRange);
            graphics.DrawRectangle(pen, rec);
            break;
        case Direction.left:
        case Direction.Left:
            rec = new Rectangle(game.PlayerLocation.X - attackRange + size, game.PlayerLocation.Y, attackRange, size);
            graphics.DrawRectangle(pen, rec);
            break;
    }
}
```

```
public bool Nearby(Direction dir, Point targetLocation, Point playerLocation, int attackRange)
{
    int size = 30; // Abs값
    switch (dir)
    {
        case Direction.up:
        case Direction.Up:
            if (
                (playerLocation.Y - targetLocation.Y) < attackRange
                && 0 < (playerLocation.Y - targetLocation.Y)
                && Math.Abs(playerLocation.X - targetLocation.X) < size
                )
                return true;
            break;
        case Direction.right:
        case Direction.Right:
            if ((playerLocation.X - targetLocation.X) > -attackRange
                && 0 > (playerLocation.X - targetLocation.X)
                && Math.Abs(playerLocation.Y - targetLocation.Y) < size
                )
                return true;
            break;
        case Direction.Down:
            if ((playerLocation.Y - targetLocation.Y) > -attackRange
                && 0 > (playerLocation.Y - targetLocation.Y)
                && Math.Abs(playerLocation.X - targetLocation.X) < size
                )
                return true;
            break;
        case Direction.left:
        case Direction.Left:
             if ((playerLocation.X - targetLocation.X) < attackRange
                 && 0 < (playerLocation.X - targetLocation.X)
                 && Math.Abs(playerLocation.Y - targetLocation.Y) < size
                 )
                 return true;
                 break;
    }
    return false;
}
```

```
protected bool DamageEnemy(Direction dir, int attackRange, int damage, Random random)
{
    Point WeaponLocation = game.PlayerLocation;//??

    for(int distance = 0; distance < attackRange; distance++)
    {
        foreach (Enemy enemy in game.Enemies)
        {
            if (Nearby(dir, enemy.Location, WeaponLocation, attackRange) && !enemy.Dead)
            {
                enemy.Hit(damage, random);
                return true;
            }
        }
        //WeaponLocation = Move(dir, game.Boundaries);  ??
    }
    return false;
}
```

sword

```
/*
 * 정면 > 시계방향 > 반시계방향 순으로 공격 체크
 * 공격반경 10
 * 데미지 3
 */
class Sword : Weapon
{
    private int attackRange = 40; // size + 10
    private int damage = 3;

        public Sword(Game game, Point location)
        : base(game, location) { }

    public override string Name { get { return "Sword"; } }
    public override void Attack(Direction dir, Random random, Form1 form)
    {
        showAttackRange(dir, form);

        if (DamageEnemy(dir, attackRange, damage, random)) return;
        else if (DamageEnemy(dir + 1, attackRange, damage, random)) return;
        else if (DamageEnemy(dir - 1, attackRange, damage, random)) return;

    }

    public override void showAttackRange(Direction dir, Form1 form)
    {
        drawAttackRange(dir, attackRange, form);
        drawAttackRange(dir + 1, attackRange, form);
        drawAttackRange(dir - 1, attackRange, form);
        System.Threading.Thread.Sleep(150);
        form.Refresh();
    }
```

potion

```
namespace TheQuest
{
    class Potion_Red: Weapon, IDrinkable
    {
        // hp 10 회복
        public Potion_Red(Game game, Point location)
            : base(game, location) { }

        public override string Name { get { return "RedPotion"; } }
        public override void Attack(Direction dir, Random random, Form1 form)
        {
            game.IncreasePlayerHealth(10, random);
        }

        public override void showAttackRange(Direction dir, Form1 form) { }
        public bool Used{ get{ return true; }}
    }
}
```

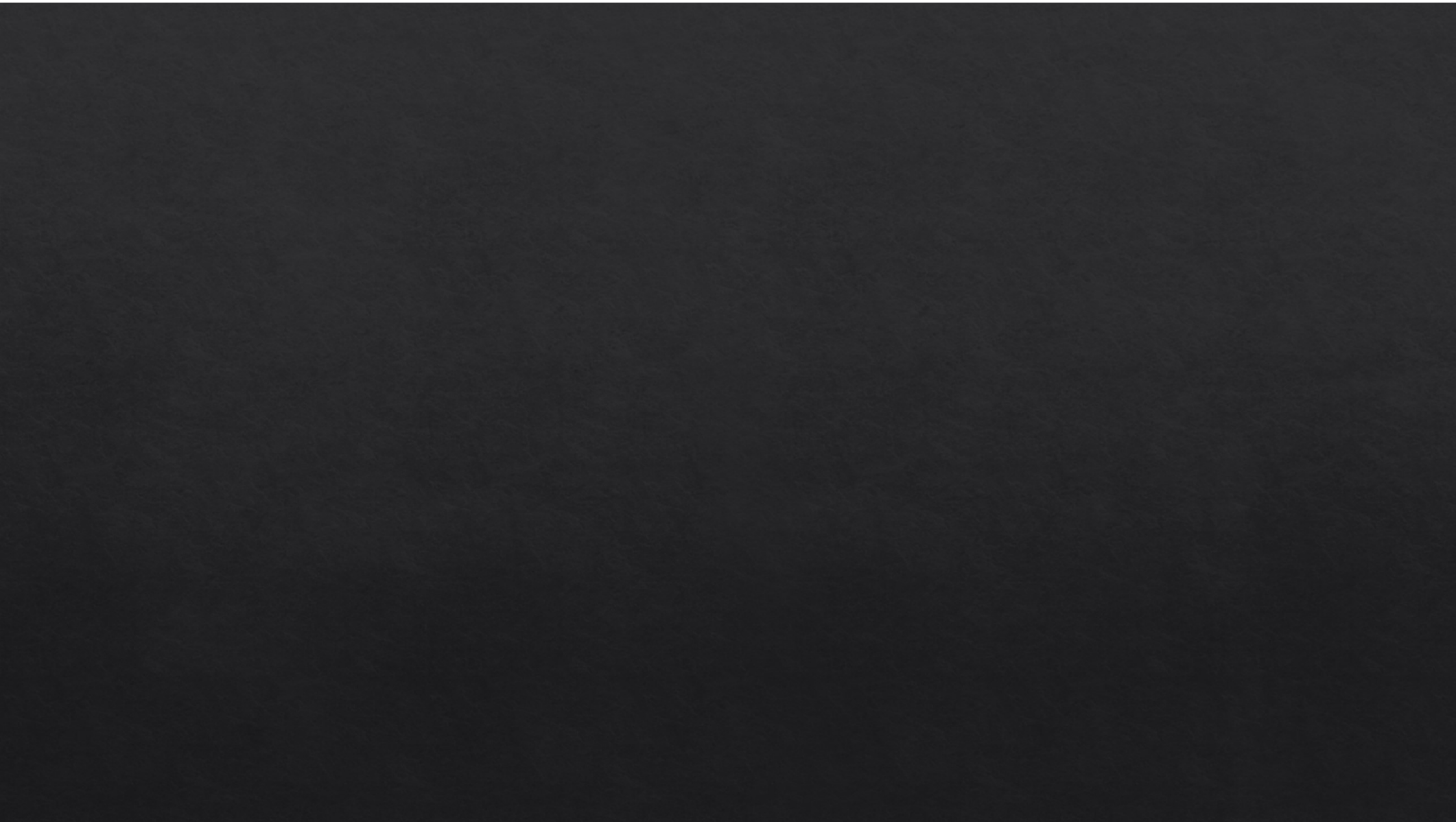
bow

```
namespace TheQuest
{
    /*
     * 정면
     * 공격반경 30
     * 데미지 1
     */
    class Bow: Weapon
    {
        private int attackRange = 60; // 30 + 30
        private int damage = 1;

        public Bow(Game game, Point location)
            : base(game, location) { }

        public override string Name { get { return "Bow"; } }
        public override void Attack(Direction dir, Random random, Form1 form)
        {
            showAttackRange(dir, form);
            DamageEnemy(dir, attackRange, damage, random);
        }

        public override void showAttackRange(Direction dir, Form1 form)
        {
            drawAttackRange(dir, attackRange, form);
            System.Threading.Thread.Sleep(150);
            form.Refresh();
        }
    }
}
```

mace

```
/*
 * 4방향 전체공격
 * 공격범위 20
 * 데미지 6
 */
class Mace : Weapon
{
    private int attackRange = 50; // size + 20
    private int damage = 6;

    public Mace(Game game, Point location)
        : base(game, location) { }

    public override string Name { get { return "Mace"; } }
    public override void Attack(Direction dir, Random random, Form1 form)
    {
        showAttackRange(dir, form);

        if(DamageEnemy(dir, attackRange, damage, random)) return;
        if (DamageEnemy(dir + 1, attackRange, damage, random)) return;
        if (DamageEnemy(dir + 2, attackRange, damage, random)) return;
        if (DamageEnemy(dir - 2, attackRange, damage, random)) return;
    }

    public override void showAttackRange(Direction dir, Form1 form)
    {
        drawAttackRange(dir, attackRange, form);
        drawAttackRange(dir + 1, attackRange, form);
        drawAttackRange(dir + 2, attackRange, form);
        drawAttackRange(dir - 2, attackRange, form);
        System.Threading.Thread.Sleep(150);
        form.Refresh();
    }
```

game.class



```
enum Direction
{
    left, // sword용
    Up,
    Right,
    Down,
    Left,
    up, // sword용
    right // mace용
}
```

Mace.class

```
if(DamageEnemy(dir, attackRange, damage, random)) return;
if (DamageEnemy(dir + 1, attackRange, damage, random)) return;
if (DamageEnemy(dir + 2, attackRange, damage, random)) return;
if (DamageEnemy(dir - 2, attackRange, damage, random)) return;
```

```
public bool Move(Direction dir, Random randomDirection)
{
    player.Move(dir);
    bool hitEffect = false;
    foreach (Enemy enemy in Enemies)
    {
        if(enemy.Move(randomDirection)) hitEffect = true;
    }
    return hitEffect;
}

public bool Attack(Direction dir, Random randomDirection, Form1 form)
{
    player.Attack(dir,randomDirection, form);
    bool hitEffect = false;
    foreach (Enemy enemy in Enemies)
    {
        if (enemy.Move(randomDirection)) hitEffect = true;
    }
    return hitEffect;
}
```

# Player.class

```
public void Attack(Direction dir,Random random, Form1 form)
{
    if (myWeapon == null) return;
    myWeapon.Attack(dir, random, form);

    if (myWeapon is IDrinkable)
    {
        inventory.Remove(myWeapon);
        if (!game.CheckPlayerInventory(myWeapon.Name)) myWeapon = null;
        else Equip(myWeapon.Name); // 다시 포션 장비
    }
}
```