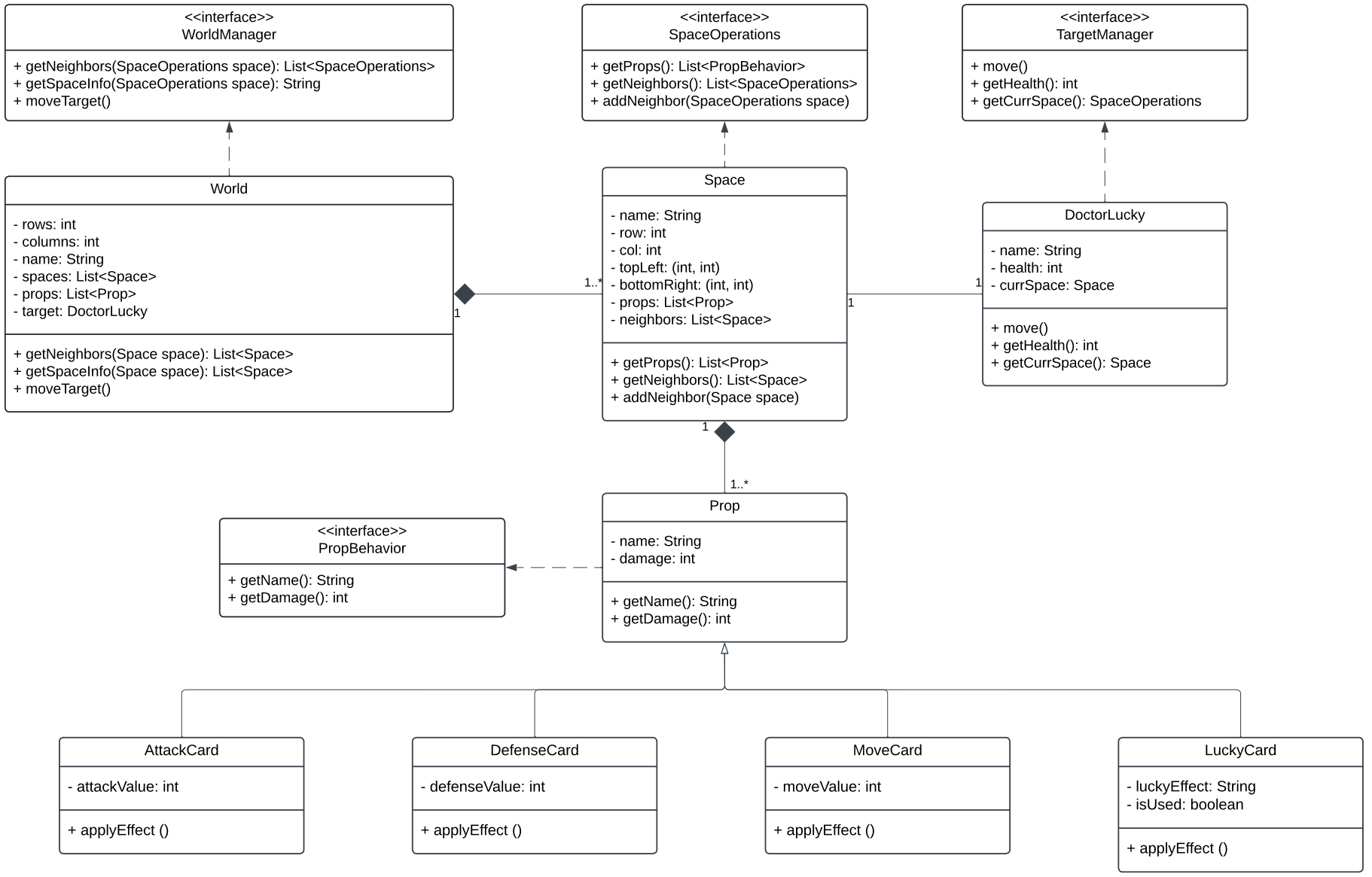
<<interface>>
WorldManager
+ getNeighbors(SpaceOperations space): List<SpaceOperations>
+ getSpaceInfo(SpaceOperations space): String
+ moveTarget()
<<interface>>
SpaceOperations
+ getProps(): List<PropBehavior>
+ getNeighbors(): List<SpaceOperations>
+ addNeighbor(SpaceOperations space)
<<interface>>
TargetManager
+ move()
+ getHealth(): int
+ getCurrSpace(): SpaceOperations
World
- rows: int
- columns: int
- name: String
- spaces: List<Space>
- props: List<Prop>
- target: DoctorLucky
+ getNeighbors(Space space): List<Space>
+ getSpaceInfo(Space space): List<Space>
+ moveTarget()
Space
- name: String
- row: int
- col: int
- topLeft: (int, int)
- bottomRight: (int, int)
- props: List<Prop>
- neighbors: List<Space>
+ getProps(): List<Prop>
+ getNeighbors(): List<Space>
+ addNeighbor(Space space)
DoctorLucky
- name: String
- health: int
- currSpace: Space
+ move()
+ getHealth(): int
+ getCurrSpace(): Space
1
1..*
1
1
1
1..*
<<interface>>
PropBehavior
+ getName(): String
+ getDamage(): int
Prop
- name: String
- damage: int
+ getName(): String
+ getDamage(): int
AttackCard
- attackValue: int
+ applyEffect ()
DefenseCard
- defenseValue: int
+ applyEffect ()
MoveCard
- moveValue: int
+ applyEffect ()
LuckyCard
- luckyEffect: String
- isUsed: boolean
+ applyEffect ()

Testing Plan

World Class

getNeighbors()

Given a space, test to see if the correct neighbor space is returned

Input

```
Space lancasterRoom = new Space("LancasterRoom", 0, 0, new int[]{0, 0}, new int[]{1, 1});
Space lilacRoom = new Space("LilacRoom", 1, 1, new int[]{1, 1}, new int[]{2, 2});
World world = new World(3, 3, "TestWorld", List.of(lancasterRoom, lilacRoom));

world.getNeighbors(lancasterRoom);
```

Expected Values

```
[
  {
    name: "LilacRoom",
    row: 1,
    col: 1,
    topLeft: (1, 1),
    bottomRight: (2, 2),
    props: [...],
    neighbors: [...]
  }
]
```

getSpaceInfo()

Getting the details of a space

Input

```
Space trophyRoom = new Space("TrophyRoom", 0, 0, new int[]{0, 0}, new int[]{1,
1});

world.getSpaceInfo(trophyRoom);
```

Expected Result

```
[
  {
  name: 'TrophyRoom',
  row: 0,
  col: 0,
  topLeft: (0, 0),
  bottomRightL (1, 1),
  props: [...],
  neighbors: [...]
  }
]
```

moveTarget()

Whether the target character moved correctly to the next space

Input

```
Space hedgeMaze = new Space("HedgeMaze", 0, 0, new int[]{0, 0}, new int[]{1,
1});
Space greenHouse = new Space("GreenHouse", 1, 1, new int[]{1, 1}, new int[]{2,
2});
World world = new World(3, 3, "TestWorld", List.of(hedgeMaze, greenHouse));
DoctorLucky target = new DoctorLucky("DoctorLucky", 100, hedgeMaze);

world.moveTarget();
target.getSpaceInfo();
```

Expected Result

```
[
  {
  name: 'GreenHouse',
  row: 1,
  col: 1,
  topLeft: (1, 1),
  bottomRightL (2, 2),
  props: [...],
  neighbors: [...]
  }
]
```

Space Class

getProps()

Get all the props in the space and make sure the props are returned correctly

Input

```
Prop attackCard = new AttackCard(50);
Prop defenseCard = new DefenseCard(30);
Space kitchen = new Space("Kitchen", 0, 0, new int[]{0, 0}, new int[]{1, 1},
List.of(attackCard, defenseCard));

kitchen.getProps();
```

Expected Result

`2`

```
[
  {
    name: "AttackCard",
    arrackValue: 20,
  },
  {
    name: "DefenseCard",
    defenseValue: 30,
  }
]
```

getNeighbors()

Test that the neighbors of the space are returned correctly

Input

```
Space diningHall = new Space("DiningHall", 0, 0, new int[]{0, 0}, new int[]{1,
1});
Space billiardRoom = new Space("BillardRoom", 1, 1, new int[]{1, 1}, new int[]
{2, 2});
space1.addNeighbor(space2);

space1.getNeighbors();
```

Expected Result

```
[
  {
    name: "BillardRoom",
    row: 1,
    col: 1,
    topLeft: (1, 1),
    bottomRight: (2, 2),
    props: [...],
    neighbors: [...]
  }
]
```

addNeighbor()

Adding a neighbor to the current space

Input

```
Space drawingRoom = new Space("DrawingRoom", 0, 0, new int[]{0, 0}, new int[]{1,
1});
Space foyer = new Space("Forer", 1, 1, new int[]{1, 1}, new int[]{2, 2});
drawingRoom.addNeighbor(foyer);

drawingRoom.getNeighbors();
```

Expected Result

```
[
  {
    name: "Forer",
    row: 1,
    col: 1,
    topLeft: (1, 1),
    bottomRight: (2, 2),
    props: [...],
    neighbors: [...]
  }
]
```

DoctorLucky Class

move()

Test DoctorLucky's movement between spaces

Input

```
Space lancasterRoom = new Space("LancasterRoom", 0, 0, new int[]{0, 0}, new
int[]{1, 1});
Space lilacRoom = new Space("LilacRoom", 1, 1, new int[]{1, 1}, new int[]{2,
2});
DoctorLucky target = new DoctorLucky("DoctorLucky", 100, lancasterRoom);
target.move(lilacRoom);

target.getCurrSpace();
```

Expected Result

```
{
  name: "LilacRoom",
  row: 1,
  col: 1,
  topLeft: (1, 1),
  bottomRight: (2, 2),
  props: [...],
  neighbors: [...]
}
```

getHealth()

Get DoctorLucky's current life value

Input

```
Space parlor = new Space("Parlor", 0, 0, new int[]{0, 0}, new int[]{1, 1});
DoctorLucky doctorLucky = new DoctorLucky("DoctorLucky", 100, parlor);

target.getHealth();
```

Expected Result

`100`

getCurrSpace()

Get DoctorLucky's current space

Input

```
Space library = new Space("Library", 0, 0, new int[]{0, 0}, new int[]{1, 1});
DoctorLucky doctorLucky = new DoctorLucky("DoctorLucky", 100, library);

target.getCurrSpace();
```

Expected Result

```
{
  name: "Library",
  row: 0,
  col: 0,
  topLeft: (0, 0),
  bottomRight: (1, 1),
  props: [...],
  neighbors: [...]
}
```

Prop Class

getName()

Get current prop name

Input

```
Prop attackCard = new Prop("AttackCard", 20);

attackCard.getName();
```

Expected Result

`AttackCard`

getDamage()

Get current prop damage value

Input

```
Prop attackCard = new Prop("AttackCard", 20);

attackCard.getDamage();
```

Expected Result

`20`

AttackCard - applyEffect()

Given a space, test to see if the correct neighbor space is returned

Input

```
Space library = new Space("Library", 0, 0, new int[]{0, 0}, new int[]{1, 1});
DoctorLucky doctorLucky = new DoctorLucky("DoctorLucky", 100, library);
AttackCard attackCard = new AttackCard(20);

attackCard.applyEffect(player);
doctorLuck.getHealth();
```

Expected Result

`80`

DefenseCard - applyEffect()

Given a space, test to see if the correct neighbor space is returned

Input

```
Space library = new Space("Library", 0, 0, new int[]{0, 0}, new int[]{1, 1});
DoctorLucky doctorLucky = new DoctorLucky("DoctorLucky", 20, library);
DefenseCard defenseCard = new DefenseCard(20);

defenseCard.applyEffect(player);
doctorLuck.getHealth();
```

Expected Result

`40`

MoveCard - applyEffect()

Given a space, test to see if the correct neighbor space is returned

Input

```
Space lilacRoom = new Space("LilacRoom", 0, 0, new int[]{0, 0}, new int[]{1,
1});
Space masterSuite = new Space("MasterSuite", 1, 1, new int[]{1, 1}, new int[]{2,
2});
DoctorLucky doctorLuck = new DoctorLucky("DoctorLucky", 100, lilacRoom);

MoveCard moveCard = new MoveCard(1);
moveCard.applyEffect(doctorLuck, masterSuite);
doctorLuck.getCurrSpace();
```

Expected Result

`masterSuite`

```
{
  name: "MasterSuite",
  row: 1,
  col: 1,
  topLeft: (1, 1),
  bottom: (2, 2),
  props: [...],
  neighbors: [...]
}
```

LuckyCard - applyEffect()

Given a space, test to see if the correct neighbor space is returned

Input

```
Space lilacRoom = new Space("LilacRoom", 0, 0, new int[]{0, 0}, new int[]{1,
1});
DoctorLucky doctorLuck = new DoctorLucky("DoctorLucky", 100, lilacRoom);

if (doctorLuck.isUsed) {
  luckyCard.applyEffect(doctorLuck);
  System.out.println("Success");
}
```

Expected Result

`Success`