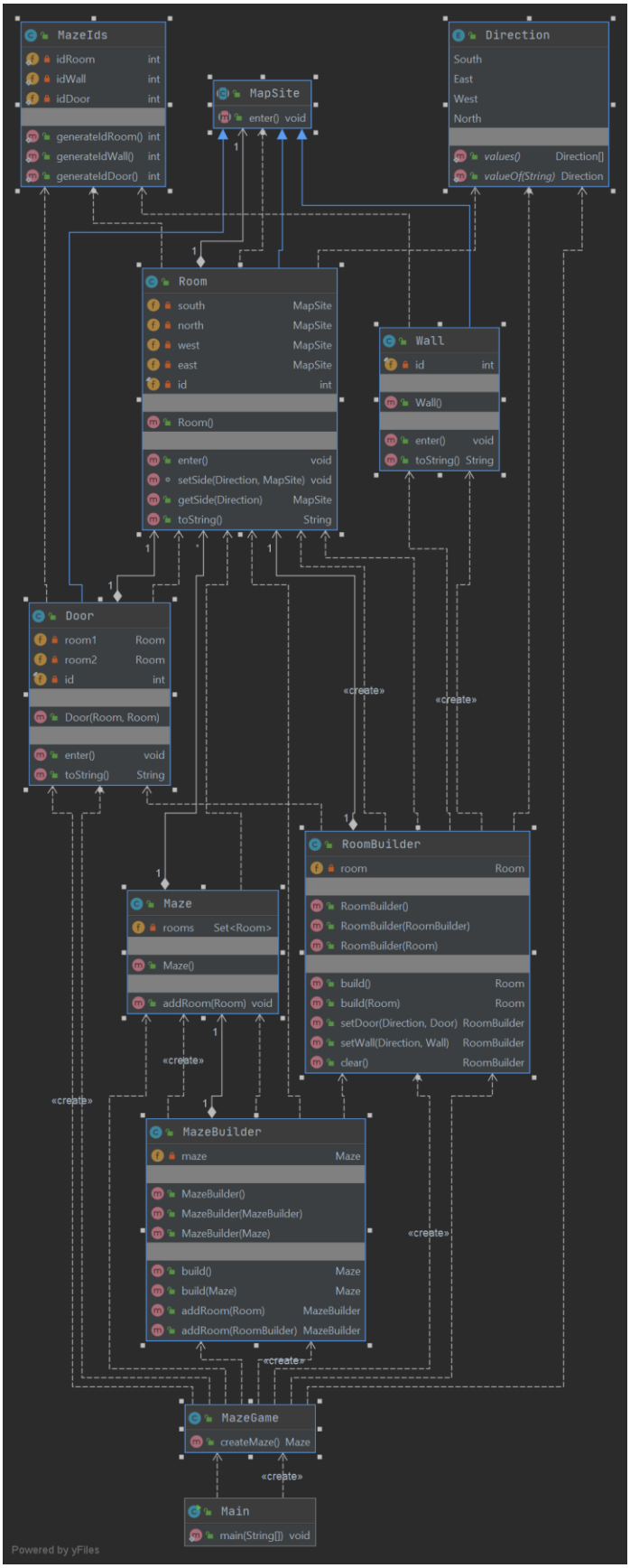
MazeIds
idRoom int
idWall int
idDoor int
generateIdRoom() int
generateIdWall() int
generateIdDoor() int
MapSite
enter() void
Direction
South
East
West
North
values() Direction[]
valueOf(String) Direction
Room
south MapSite
north MapSite
west MapSite
east MapSite
id int
Room()
enter() void
setSide(Direction, MapSite) void
getSide(Direction) MapSite
toString() String
Wall
id int
Wall()
enter() void
toString() String
Door
room1 Room
room2 Room
id int
Door(Room, Room)
enter() void
toString() String
RoomBuilder
room Room
RoomBuilder()
RoomBuilder(RoomBuilder)
RoomBuilder(Room)
build() Room
build(Room) Room
setDoor(Direction, Door) RoomBuilder
setWall(Direction, Wall) RoomBuilder
clear() RoomBuilder
Maze
rooms Set<Room>
Maze()
addRoom(Room) void
MazeBuilder
maze Maze
MazeBuilder()
MazeBuilder(MazeBuilder)
MazeBuilder(Maze)
build() Maze
build(Maze) Maze
addRoom(Room) MazeBuilder
addRoom(RoomBuilder) MazeBuilder
MazeGame
createMaze() Maze
Main
main(String[]) void
«create»
1
Powered by yFiles

```java
public enum Direction {
    South, East, West, North
}
```

```java
public abstract class MapSite {
    public abstract void enter();
}
```

```java
public class Wall extends MapSite {
    private final int id;
    {
        id = MazeIds.generateIdWall();
    }

    public Wall() {
        System.out.println("constructor wall "+id);
    }

    @Override
    public void enter() {
        System.out.println("can not enter in wall ");
    }

    @Override
    public String toString() {
        return "Wall{" +
                "id=" + id +
                '}';
    }
}
```

```java
public class Door extends  MapSite {
    private Room room1;
    private Room room2;
    private final int id;
    {
        id = MazeIds.generateIdDoor();
    }

    public Door(Room room1, Room room2) {
        this.room1 = room1;
        this.room2 = room2;
        System.out.println("construct door id " + id +
                "between " + room1 + " and " + room2);
    }

    @Override
    public void enter() {
        System.out.println();
    }

    @Override
```

```
    public String toString() {
        return "Door{" +
                "id=" + id +
                '}';
    }
}
```

```
public class Room extends MapSite {
    private MapSite south;
    private MapSite north;
    private MapSite west;
    private MapSite east;
    private final int id;
    {
        id = MazeIds.generateIdRoom();
    }

    public Room() {
        System.out.println("constructor room "+id);
    }

    @Override
    public void enter() {
        System.out.println("Enter in room");
    }

    void setSide(Direction d, MapSite site){

        if (d == Direction.North){
            north = site;
        } else if (d == Direction.South){
            south = site;
        } else if (d == Direction.East){
            east = site;
        } else if (d == Direction.West){
            west = site;
        }
        System.out.println("set " + d.toString() +
                " side of " + this.toString() + " to " +
                site.toString());
    }

    public MapSite getSide(Direction d){
        MapSite result = null;
        if (d == Direction.North){
            result = north;
        } else if (d == Direction.South){
            result = south;
        } else if (d == Direction.East){
            result = east;
        } else if (d == Direction.West){
            result = west;
        }
        return result;
```

```
    }

    @Override
    public String toString() {
        return "Room{" +
                "nr=" + id +
                '}';
    }
}
```

```
public class MazeIds {
    private static int idRoom = 1;
    private static int idWall = 1;
    private static int idDoor = 1;

    public static int generateIdRoom(){
        int tmp = idRoom;
        idRoom++;
        return tmp;
    }

    public static int generateIdWall(){
        int tmp = idWall;
        idWall++;
        return tmp;
    }

    public static int generateIdDoor(){
        int tmp = idDoor;
        idDoor++;
        return tmp;
    }
}
```

```
import java.util.HashSet;
import java.util.Set;

public class Maze {
    private Set<Room> rooms = new HashSet<>();
    public Maze() {
        System.out.println("maze constructor");
    }

    public void addRoom(Room room){
        if(rooms.contains(room)){
            return;
        }
        rooms.add(room);
    }
}
```

```
public class MazeBuilder {
   private Maze maze;

   public MazeBuilder() {
      maze = new Maze();
   }

   public MazeBuilder(MazeBuilder mazeBuilder) {
      maze = mazeBuilder.build();
   }
   public MazeBuilder(Maze maze) {
      this.maze = maze;
   }

   public Maze build(){
      return maze;
   }

   public Maze build(Maze maze){
      return maze;
   }

   public MazeBuilder addRoom(Room room){
      maze.addRoom(room);
      return this;
   }

   public MazeBuilder addRoom(RoomBuilder roomBuilder){
      maze.addRoom(roomBuilder.build());
      return this;
   }
}
```

```
public class RoomBuilder {
   private Room room;

   public RoomBuilder() {
      room = new Room();
      room.setSide(Direction.East, new Wall());
      room.setSide(Direction.West, new Wall());
      room.setSide(Direction.North, new Wall());
      room.setSide(Direction.South, new Wall());
   }

   public RoomBuilder(RoomBuilder roomBuilder) {
      room = roomBuilder.build();
   }
   public RoomBuilder(Room room) {
      this.room = room;
   }

   public Room build(){
      return room;
   }
```

```
    public Room build(Room room){
        return room;
    }

    public RoomBuilder setDoor(Direction d,Door door){
        room.setSide(d,door);
        return this;
    }

    public RoomBuilder setWall(Direction d,Wall wall){
        room.setSide(d,wall);
        return this;
    }

    public RoomBuilder clear(){
        room = new Room();
        room.setSide(Direction.East, new Wall());
        room.setSide(Direction.West, new Wall());
        room.setSide(Direction.North, new Wall());
        room.setSide(Direction.South, new Wall());
        return this;
    }
}
```

```
public class MazeGame {
    public Maze createMaze(){
        MazeBuilder mazeBuilder = new MazeBuilder();
        RoomBuilder r1 = new RoomBuilder();
        RoomBuilder r2 = new RoomBuilder();
        Door door = new Door(r1.build(),r2.build());

        return mazeBuilder.addRoom(r1.setDoor(Direction.East, door))
                .addRoom(r2.setDoor(Direction.West, door)).build();
//        Maze aMaze = new Maze();
//        Room r1 = new Room();
//        Room r2 = new Room();
//        Door theDoor = new Door(r1,r2);
//        aMaze.addRoom(r1); aMaze.addRoom(r2);
//        r1.setSide(Direction.North, new Wall());
//        r1.setSide(Direction.East, theDoor);
//        r1.setSide(Direction.South, new Wall());
//        r1.setSide(Direction.West, new Wall());
//        r2.setSide(Direction.North, new Wall());
//        r2.setSide(Direction.East, new Wall());
//        r2.setSide(Direction.South, new Wall());
//        r2.setSide(Direction.West, theDoor);
//        return aMaze;
    }


}
```

```java
public class Main {
    public static void main(String[] args) {
        MazeGame game = new MazeGame();
        game.createMaze();
    }
}
```