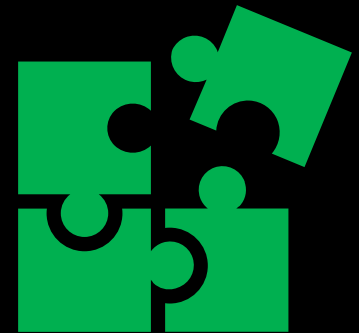
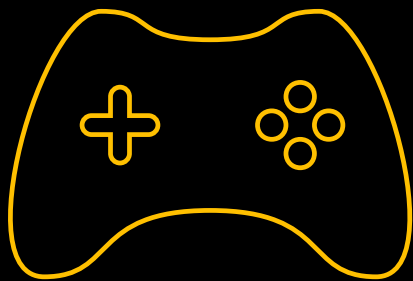


Code Schnipsel

Die lilafarben markierten Codeteile müsst ihr selbst ausfüllen.



```
public enum Direction {
    UP(new Position(0, 1), Input.Keys.W),
    DOWN(???, Input.Keys.S),
    LEFT(new Position(-1, 0), ???),
    RIGHT(???, Input.Keys.D);

    private final Position position;
    private final int key;

    Direction(Position position, int key) {
        this.position = position;
        ???
    }

    public int getKey() {
        return key;
    }

    ??? (getter für position)
}
```

```
public class Colors {
    // Die Farben bestehen aus red, green, blue und alpha (= Opazität).
    // Opazität: 0f = unsichtbar, 0.5f = semi transparent, 1f = völlig undurchsichtig.
    // Alle Komponenten werden zwischen 0f und 1f angegeben.
    public static final Color
        POSITIVE_COLOR = new Color(???, ???, ???, ???),
        NEGATIVE_COLOR = new Color(???, ???, ???, ???),
        NEUTRAL_COLOR = new Color(???, ???, ???, ???),
        HEALTH_BAR_COLOR = new Color(???, ???, ???, ???);
}

public record Position(int x, int y) {

    public Position add(Position other) {
        return new Position(???);
    }
}
```

```

public enum Field {
    NEUTRAL("dot0", 0),
    ONE_POSITIVE("dot1", 1), ???, ???, FOUR_POSITIVE("dot4", 4),
    ???, TWO_NEGATIVE("dot2", -2), ???, ???

    private final String textureName;
    private final ??? color;
    private final int points;

    Field(String textureName, int points) {
        this.textureName = textureName;
        ??? (initialisiere points)
        this.color = points < 0 ? NEGATIVE_COLOR : ??? (zweiter Ternary);
    }

    public String getTextureName() {
        return textureName + ".png";
    }

    ??? (getter für color und points)
}

```

```
public class World {
    private final HashMap<Position, Field> fields = new HashMap<>();
    private final Random random = new Random();
    private Position playerPosition = new Position(0, 0);
    private int points = 100;

    public Field getField(Position position) {
        Field field = ??? (Hole das Feld an Position position aus der HashMap fields.)
        if (field == null) {
            Field[] allFields = Field.values();
            field = ??? (Wähle ein zufälliges Feld aus.)
            // Tipp: Schau dir die Aufgabe "Lustige Sätze" an.
            fields.put(position, field);
        }
        return field;
    }

    private void move(Direction direction) {
        ??? (Schwierig: Was muss passieren, wenn man sich bewegt?)
    }
}
```

World fertigstellen

Ergänze World um folgende Methoden:

```
public void update() {  
    if (!isPlayerAlive()) return;  
    for (Direction direction : Direction.values())  
        if (Gdx.input.isKeyJustPressed(direction.getKey()))  
            move(direction);  
}
```

- Methode `isPlayerAlive` (Gibt es noch Punkte?)
- getter für `playerPosition` und `points`

Klasse Renderer

Die Klasse Renderer stellt die Welt auf dem Bildschirm da.

Weil sie sehr lang ist und nicht viele Erkenntnisse bietet, haben wir uns entschieden, euch sie zum Kopieren zu geben.

Erstellt die Klasse im Ordner `root.content`. Der Name muss wie immer korrekt geschrieben sein.

```
public class Main extends ApplicationAdapter {
    private final World world = new World();
    private Renderer renderer;

    @Override
    public void create() {
        ??? (Initialisiere renderer durch ein neues Objekt.)
    }

    @Override
    public void resize(int width, int height) {
        ??? (Rufe die resize-Methode von renderer auf.)
    }

    @Override
    public void render() {
        world.update();
        ??? (Verwende renderer, um die Welt darzustellen.)
    }
}
```


Eingabe von Symbolen

Name	Kombination	Symbole
Semikolon, Doppelpunkt	Shift + Komma (,) / Punkt (.)	; :
Anführungsstrich(e)	Shift + Hashtag (#) / 2	' "
Runde Klammern	Shift + 8/9	()
Eckige Klammern	Alt Gr + 8/9	[]
Geschweifte Klammern	Alt Gr + 7/0	{}
Ausrufe/Fragezeichen	Shift + 1/ß	! ?
Und, Gleich	Shift + 6/0	& =
Senkrechter Strich	Alt Gr + <	
Größer	Shift + <	>
Unterstrich	Shift + -	_

Tastatur-Shortcuts

Name	Kombination
Element auswählen	Strg + W
Alles auswählen	Strg + A
Kopieren	Strg + C
Einfügen	Strg + V
Löschen und Kopieren	Strg + X
Duplizieren	Strg + D
Rückgängig	Strg + Z
Rückrückgängig	Strg + Y
Suchen	Strg + F
Suchen und Ersetzen	Strg + R

IntelliJ-Shortcuts

Name

Nach oben / unten scrollen

Mauszeiger nach links / rechts bewegen

Zu Definition / Verwendung springen

Methoden überschreiben

Schnell durch das Projekt navigieren

Programm starten

Code formatieren (schön einrücken)

Zurück zum vorherigen Mauszeiger

Zu Mauszeiger springen

Kombination

Strg + ↑↓

Strg + ←→

Strg + B

Strg + O

2x Shift

Shift + F10

Strg + Alt + L

Strg + Alt + ←→

Strg + M