Software Engineering Besprechung Übung1

Programmverständnis, Dokumentation

Teaching Team

Paul Muntean
muntean@ifi.uzh.ch

Martina Rakaric
martina.rakaric@gmail.com

Abgabe

Abgabe	OLAT
Erlaubte Datentypen	PDF Datei oder .zip-File falls Source Code enthalten
Beschriftung PDF	Ex[**n**] [**NameA_Matrikelnummer**].pdf, wobei [**n**] die Nummer der Übung ist. Die PDF Datei sollte außerdem ebenfalls Ihren Namen und Matrikelnummer beinhalten.
Einteilung	Einzelarbeit

Aufgabe 1 und 2

- Aufgabe 1
  - Allgemeine Infos
- Aufgabe 2
  - Einführung und Installation

Aufgabe 3: Verstehen des Quellcodes (12 Punkte)

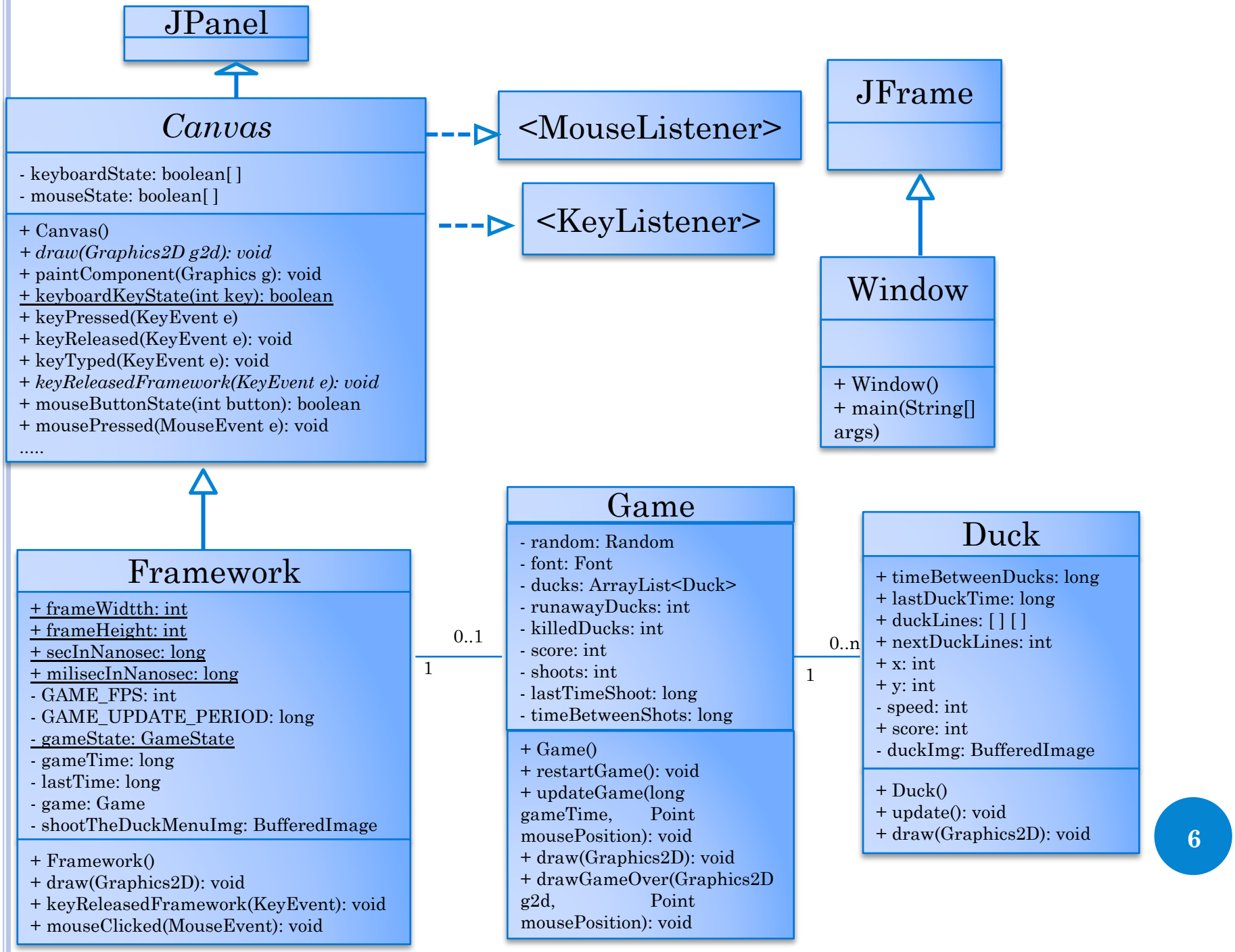
Aufgabe 3.1: Struktur

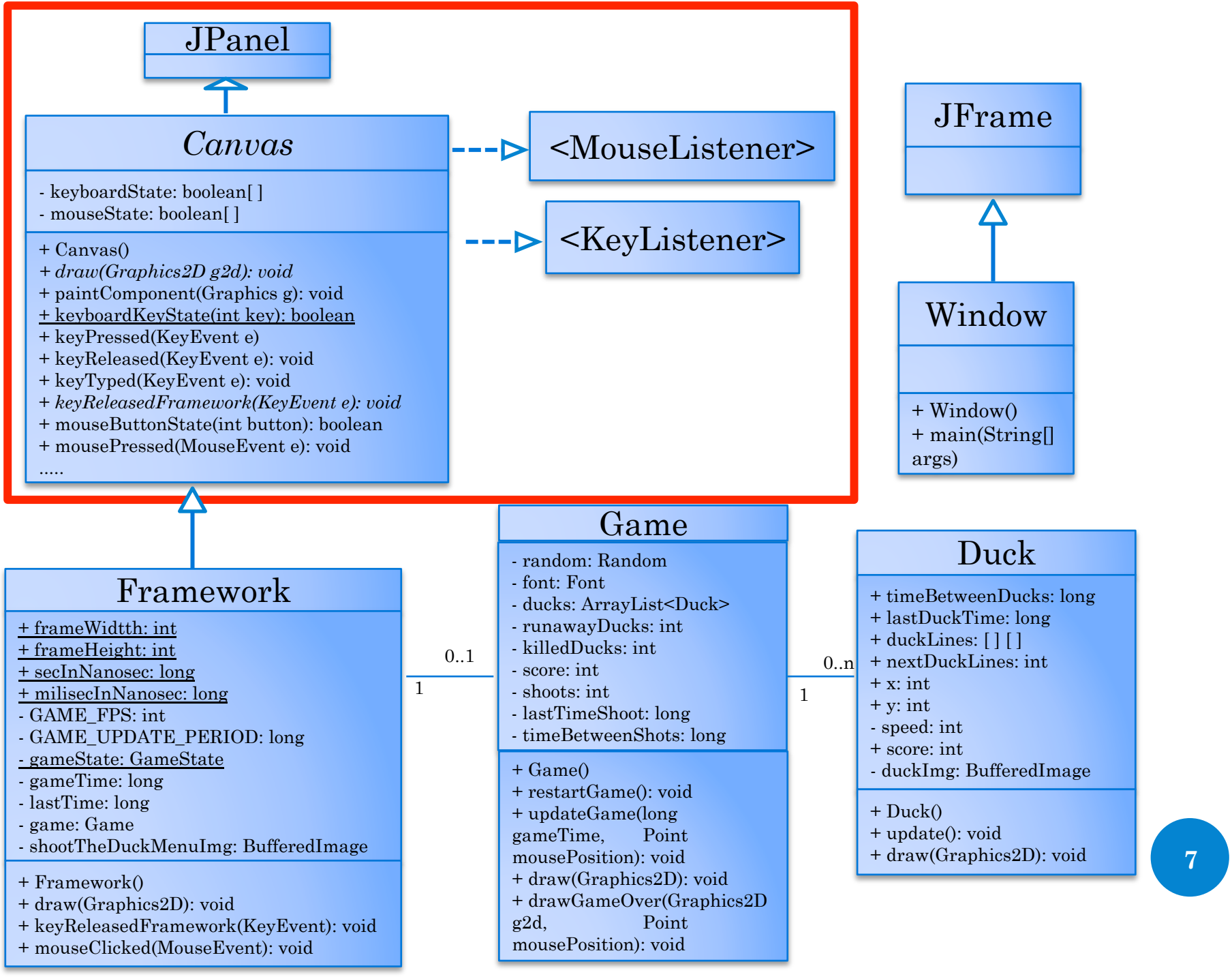
Ziel:

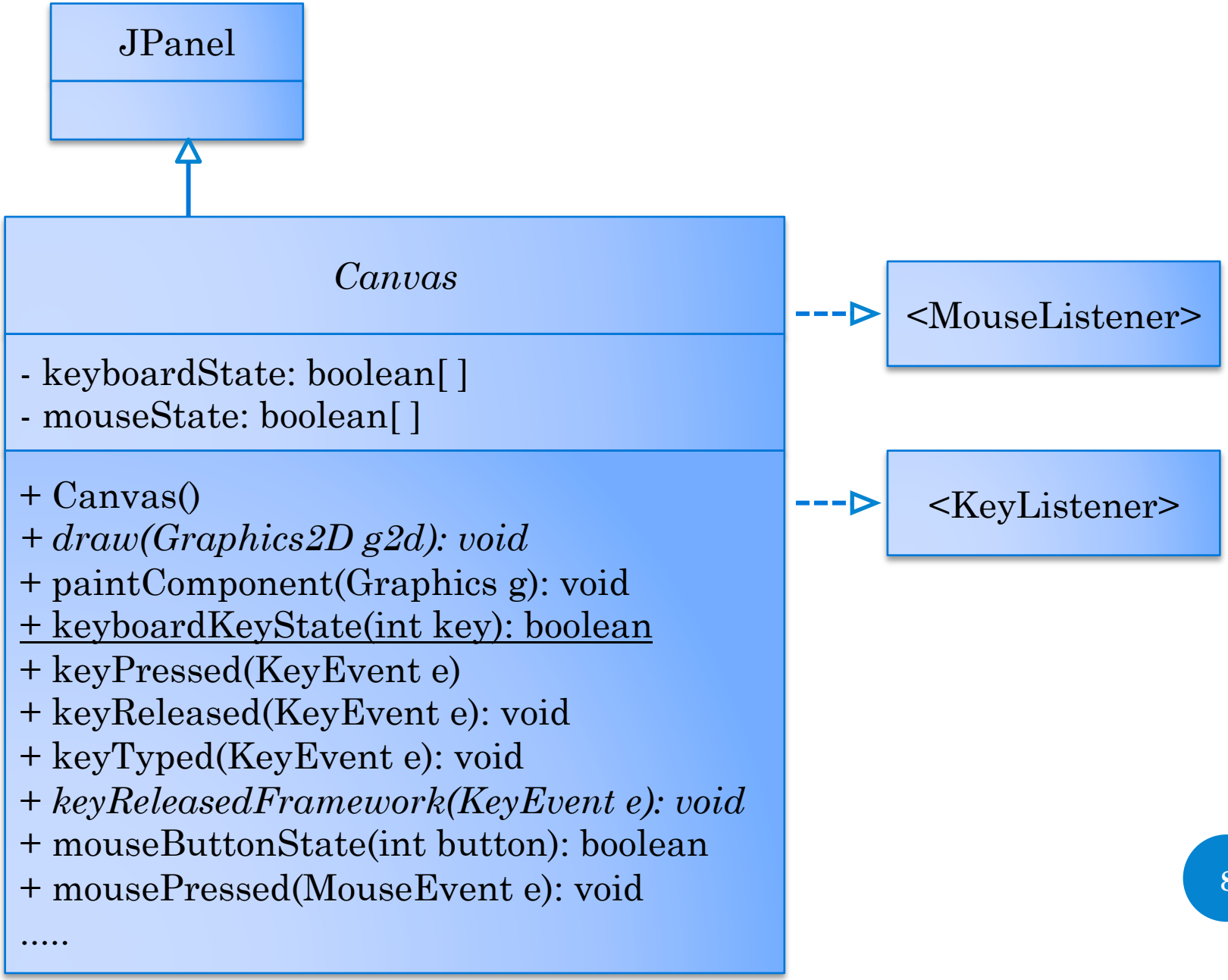
- Gesamtüberblick über Quellcode gewinnen
- UML Klassendiagramm Notationen auffrischen

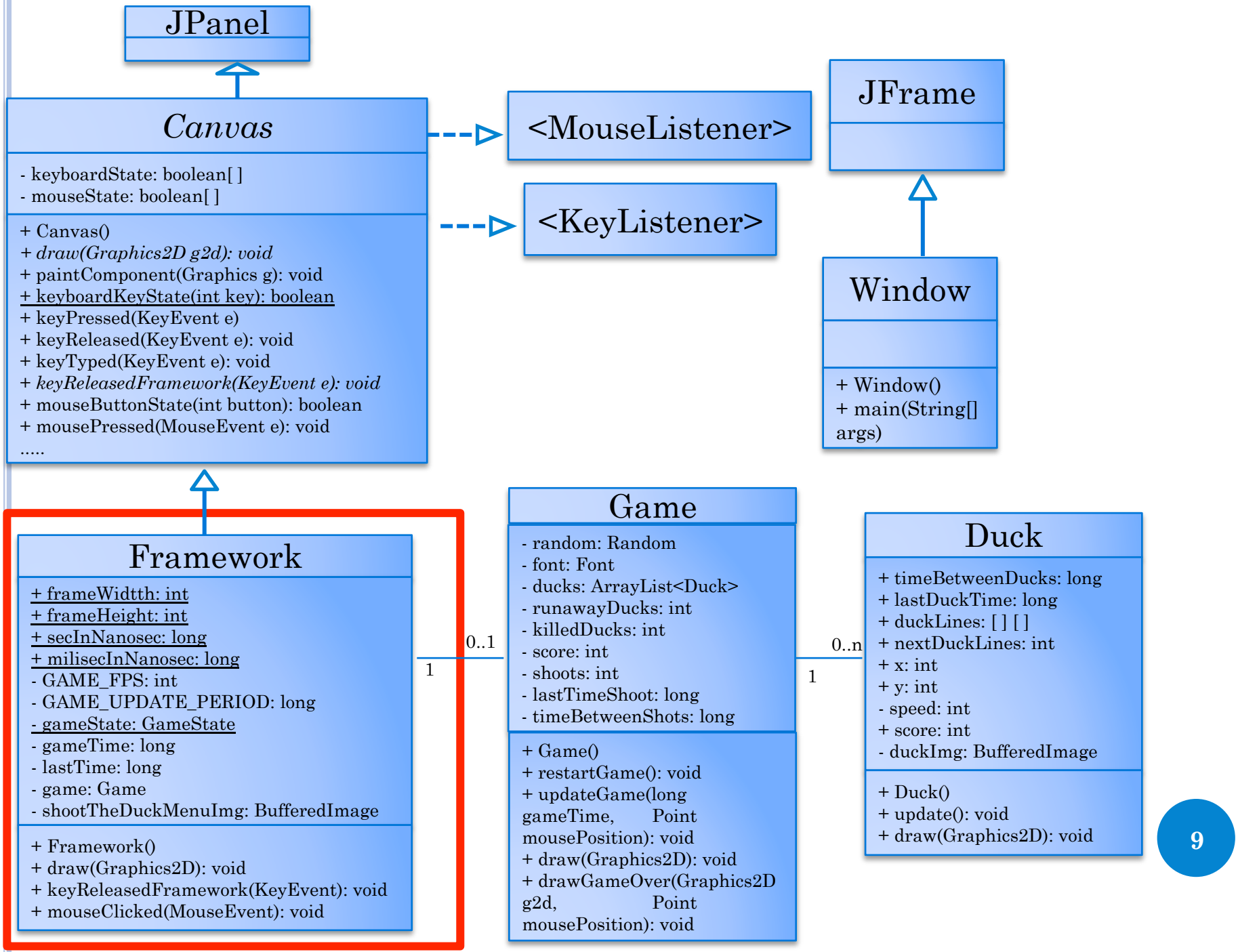
Aufgabe:

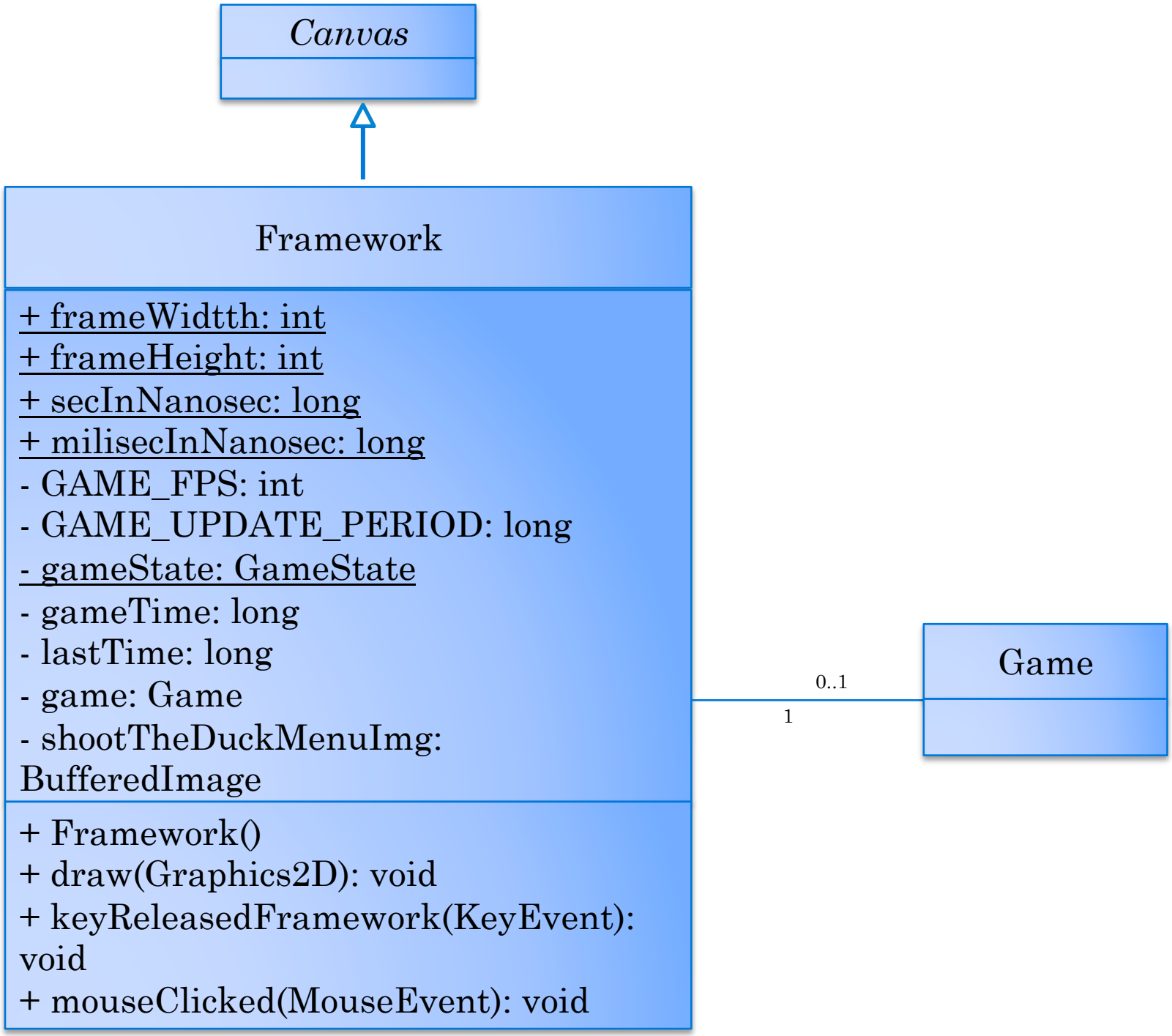
- Klassendiagramm modellieren
- Alle Klassenvariabeln & Public Methoden modellieren

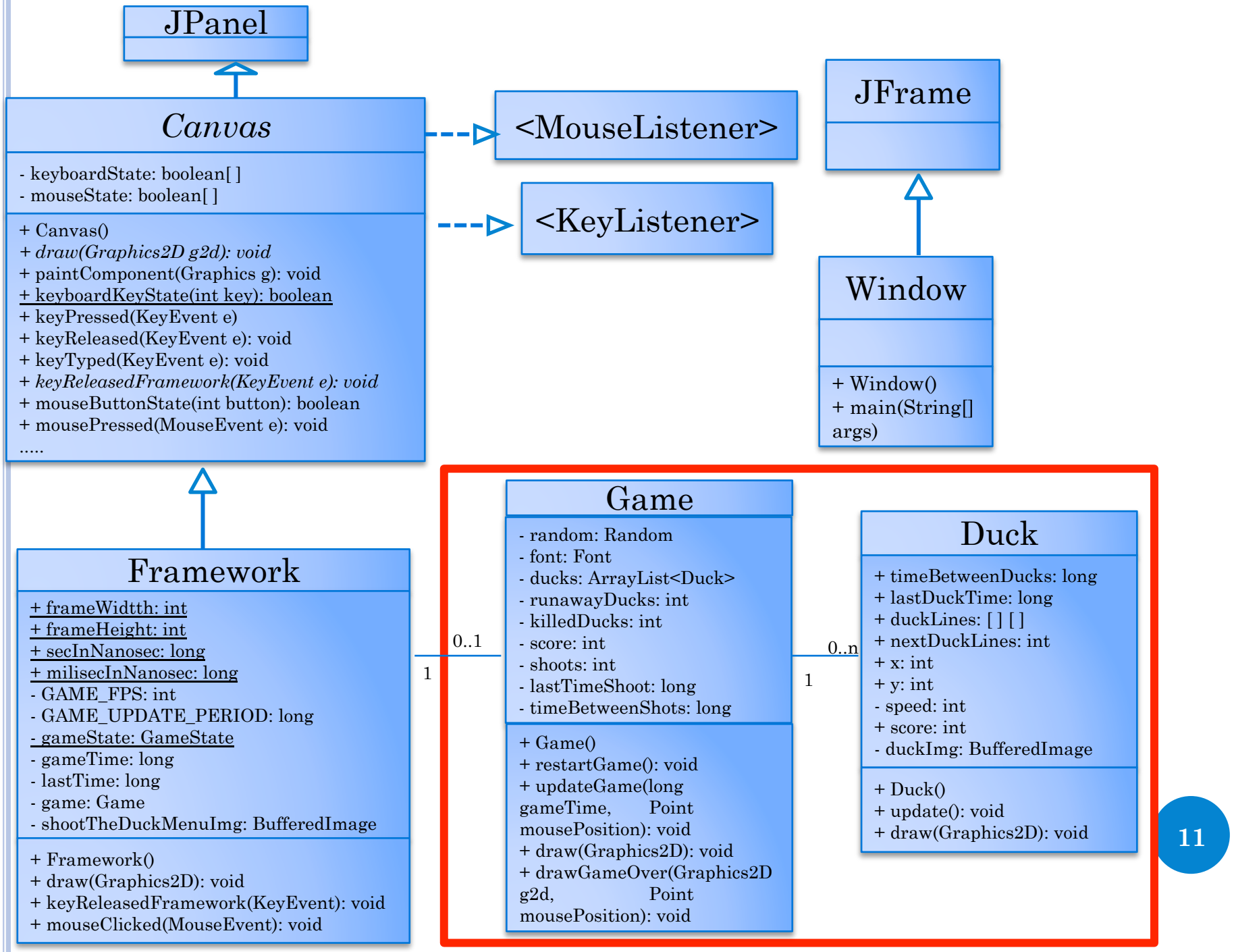

JPanel
Canvas
- keyboardState: boolean[ ]
- mouseState: boolean[ ]
+ Canvas()
+ draw(Graphics2D g2d): void
+ paintComponent(Graphics g): void
+ keyboardKeyState(int key): boolean
+ keyPressed(KeyEvent e)
+ keyReleased(KeyEvent e): void
+ keyTyped(KeyEvent e): void
+ keyReleasedFramework(KeyEvent e): void
+ mouseButtonState(int button): boolean
+ mousePressed(MouseEvent e): void
.....
<MouseListener>
<KeyListener>
JFrame
Window
+ Window()
+ main(String[] args)
Framework
+ frameWidtth: int
+ frameHeight: int
+ secInNanosec: long
+ milisecInNanosec: long
- GAME_FPS: int
- GAME_UPDATE_PERIOD: long
- gameState: GameState
- gameTime: long
- lastTime: long
- game: Game
- shootTheDuckMenuImg: BufferedImage
+ Framework()
+ draw(Graphics2D): void
+ keyReleasedFramework(KeyEvent): void
+ mouseClicked(MouseEvent): void
0..1
1
Game
- random: Random
- font: Font
- ducks: ArrayList<Duck>
- runawayDucks: int
- killedDucks: int
- score: int
- shoots: int
- lastTimeShoot: long
- timeBetweenShots: long
+ Game()
+ restartGame(): void
+ updateGame(long gameTime, Point mousePosition): void
+ draw(Graphics2D): void
+ drawGameOver(Graphics2D g2d, Point mousePosition): void
0..n
1
Duck
+ timeBetweenDucks: long
+ lastDuckTime: long
+ duckLines: [ ] [ ]
+ nextDuckLines: int
+ x: int
+ y: int
- speed: int
+ score: int
- duckImg: BufferedImage
+ Duck()
+ update(): void
+ draw(Graphics2D): void

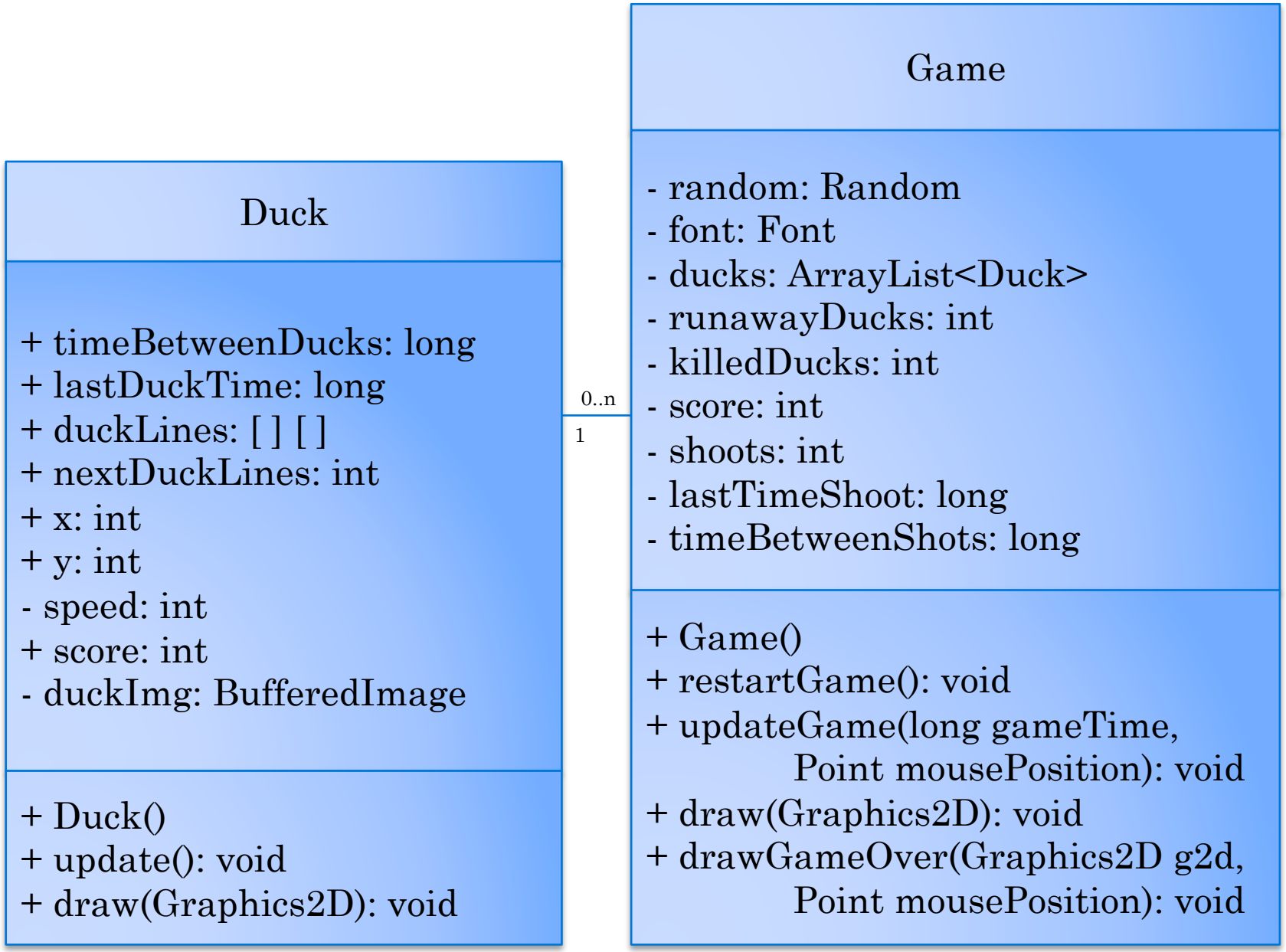

JPanel
Canvas
- keyboardState: boolean[ ]
- mouseState: boolean[ ]
+ Canvas()
+ draw(Graphics2D g2d): void
+ paintComponent(Graphics g): void
+ keyboardKeyState(int key): boolean
+ keyPressed(KeyEvent e)
+ keyReleased(KeyEvent e): void
+ keyTyped(KeyEvent e): void
+ keyReleasedFramework(KeyEvent e): void
+ mouseButtonState(int button): boolean
+ mousePressed(MouseEvent e): void
.....
<MouseListener>
<KeyListener>
JFrame
Window
+ Window()
+ main(String[] args)
Framework
+ frameWidtth: int
+ frameHeight: int
+ secInNanosec: long
+ milisecInNanosec: long
- GAME_FPS: int
- GAME_UPDATE_PERIOD: long
- gameState: GameState
- gameTime: long
- lastTime: long
- game: Game
- shootTheDuckMenuImg: BufferedImage
+ Framework()
+ draw(Graphics2D): void
+ keyReleasedFramework(KeyEvent): void
+ mouseClicked(MouseEvent): void
0..1
1
Game
- random: Random
- font: Font
- ducks: ArrayList<Duck>
- runawayDucks: int
- killedDucks: int
- score: int
- shoots: int
- lastTimeShoot: long
- timeBetweenShots: long
+ Game()
+ restartGame(): void
+ updateGame(long gameTime, Point mousePosition): void
+ draw(Graphics2D): void
+ drawGameOver(Graphics2D g2d, Point mousePosition): void
0..n
1
Duck
+ timeBetweenDucks: long
+ lastDuckTime: long
+ duckLines: [ ] [ ]
+ nextDuckLines: int
+ x: int
+ y: int
- speed: int
+ score: int
- duckImg: BufferedImage
+ Duck()
+ update(): void
+ draw(Graphics2D): void


JPanel
Canvas
- keyboardState: boolean[ ]
- mouseState: boolean[ ]
+ Canvas()
+ draw(Graphics2D g2d): void
+ paintComponent(Graphics g): void
+ keyboardKeyState(int key): boolean
+ keyPressed(KeyEvent e)
+ keyReleased(KeyEvent e): void
+ keyTyped(KeyEvent e): void
+ keyReleasedFramework(KeyEvent e): void
+ mouseButtonState(int button): boolean
+ mousePressed(MouseEvent e): void
.....
<MouseListener>
<KeyListener>

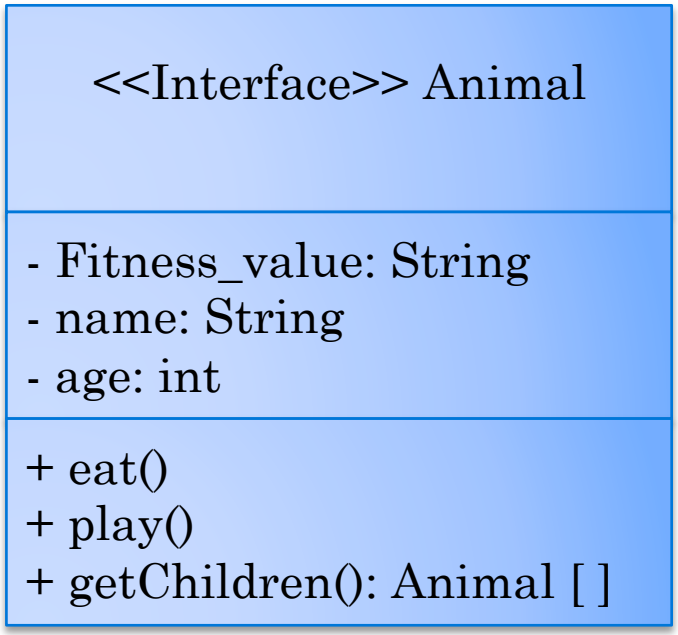

JPanel
Canvas
- keyboardState: boolean[ ]
- mouseState: boolean[ ]
+ Canvas()
+ draw(Graphics2D g2d): void
+ paintComponent(Graphics g): void
+ keyboardKeyState(int key): boolean
+ keyPressed(KeyEvent e)
+ keyReleased(KeyEvent e): void
+ keyTyped(KeyEvent e): void
+ keyReleasedFramework(KeyEvent e): void
+ mouseButtonState(int button): boolean
+ mousePressed(MouseEvent e): void
.....
<MouseListener>
<KeyListener>
JFrame
Window
+ Window()
+ main(String[] args)
Framework
+ frameWidtth: int
+ frameHeight: int
+ secInNanosec: long
+ milisecInNanosec: long
- GAME_FPS: int
- GAME_UPDATE_PERIOD: long
- gameState: GameState
- gameTime: long
- lastTime: long
- game: Game
- shootTheDuckMenuImg: BufferedImage
+ Framework()
+ draw(Graphics2D): void
+ keyReleasedFramework(KeyEvent): void
+ mouseClicked(MouseEvent): void
1
0..1
Game
- random: Random
- font: Font
- ducks: ArrayList<Duck>
- runawayDucks: int
- killedDucks: int
- score: int
- shoots: int
- lastTimeShoot: long
- timeBetweenShots: long
+ Game()
+ restartGame(): void
+ updateGame(long gameTime, Point mousePosition): void
+ draw(Graphics2D): void
+ drawGameOver(Graphics2D g2d, Point mousePosition): void
1
0..n
Duck
+ timeBetweenDucks: long
+ lastDuckTime: long
+ duckLines: [ ] [ ]
+ nextDuckLines: int
+ x: int
+ y: int
- speed: int
+ score: int
- duckImg: BufferedImage
+ Duck()
+ update(): void
+ draw(Graphics2D): void


Canvas
Framework
+ frameWidtth: int
+ frameHeight: int
+ secInNanosec: long
+ milisecInNanosec: long
- GAME_FPS: int
- GAME_UPDATE_PERIOD: long
- gameState: GameState
- gameTime: long
- lastTime: long
- game: Game
- shootTheDuckMenuImg: BufferedImage
+ Framework()
+ draw(Graphics2D): void
+ keyReleasedFramework(KeyEvent): void
+ mouseClicked(MouseEvent): void
0..1
1
Game


JPanel
Canvas
- keyboardState: boolean[ ]
- mouseState: boolean[ ]
+ Canvas()
+ draw(Graphics2D g2d): void
+ paintComponent(Graphics g): void
+ keyboardKeyState(int key): boolean
+ keyPressed(KeyEvent e)
+ keyReleased(KeyEvent e): void
+ keyTyped(KeyEvent e): void
+ keyReleasedFramework(KeyEvent e): void
+ mouseButtonState(int button): boolean
+ mousePressed(MouseEvent e): void
.....
<MouseListener>
<KeyListener>
JFrame
Window
+ Window()
+ main(String[] args)
Framework
+ frameWidtth: int
+ frameHeight: int
+ secInNanosec: long
+ milisecInNanosec: long
- GAME_FPS: int
- GAME_UPDATE_PERIOD: long
- gameState: GameState
- gameTime: long
- lastTime: long
- game: Game
- shootTheDuckMenuImg: BufferedImage
+ Framework()
+ draw(Graphics2D): void
+ keyReleasedFramework(KeyEvent): void
+ mouseClicked(MouseEvent): void
1
0..1
Game
- random: Random
- font: Font
- ducks: ArrayList<Duck>
- runawayDucks: int
- killedDucks: int
- score: int
- shoots: int
- lastTimeShoot: long
- timeBetweenShots: long
+ Game()
+ restartGame(): void
+ updateGame(long gameTime, Point mousePosition): void
+ draw(Graphics2D): void
+ drawGameOver(Graphics2D g2d, Point mousePosition): void
1
0..n
Duck
+ timeBetweenDucks: long
+ lastDuckTime: long
+ duckLines: [ ] [ ]
+ nextDuckLines: int
+ x: int
+ y: int
- speed: int
+ score: int
- duckImg: BufferedImage
+ Duck()
+ update(): void
+ draw(Graphics2D): void


Duck
+ timeBetweenDucks: long
+ lastDuckTime: long
+ duckLines: [ ] [ ]
+ nextDuckLines: int
+ x: int
+ y: int
- speed: int
+ score: int
- duckImg: BufferedImage
+ Duck()
+ update(): void
+ draw(Graphics2D): void
0..n
1
Game
- random: Random
- font: Font
- ducks: ArrayList<Duck>
- runawayDucks: int
- killedDucks: int
- score: int
- shoots: int
- lastTimeShoot: long
- timeBetweenShots: long
+ Game()
+ restartGame(): void
+ updateGame(long gameTime,
Point mousePosition): void
+ draw(Graphics2D): void
+ drawGameOver(Graphics2D g2d,
Point mousePosition): void

Kritik



+ public, - private, # protected, (~ package)

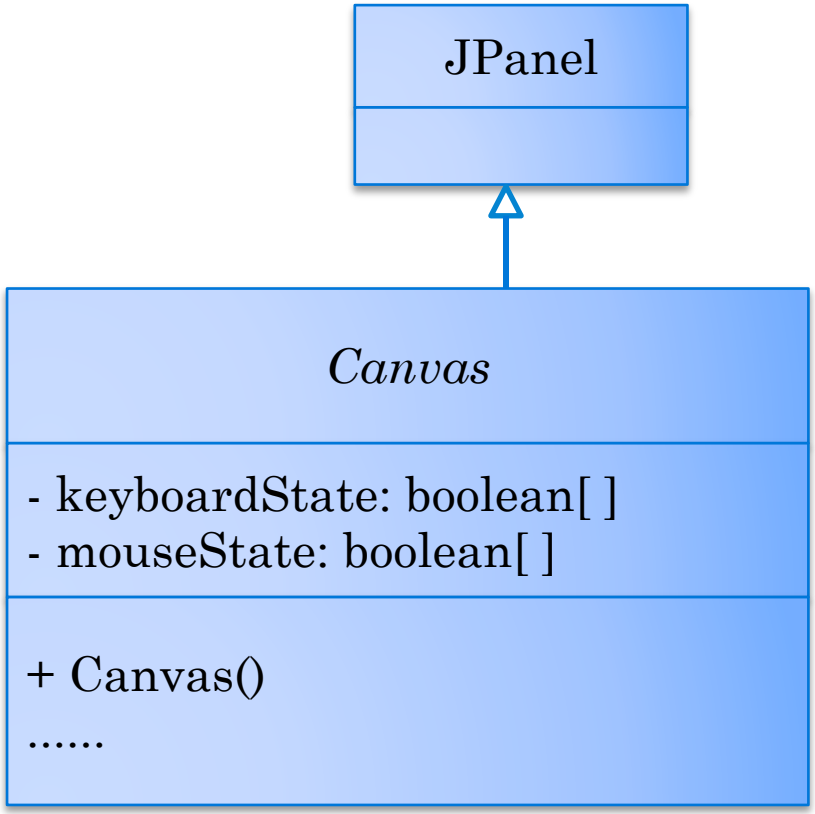
Package Name weglassen

Primitive Datentypen klein: int, double
Klassen gross: String

Array: square braces [ ]
Static: existiert nur einmal, unterstrichen
Final: unveränderbar, Grossbuchstaben

Operation:
visibility name (Parameter): Typ
Parameter:
Name: Typ

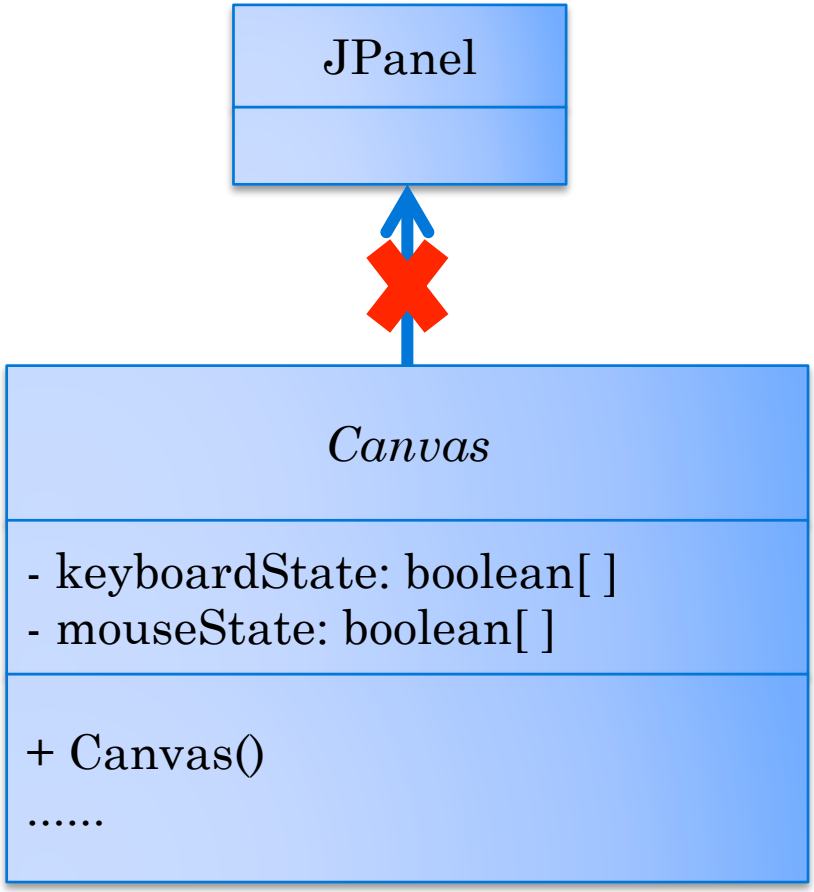
Kritik Vererbung



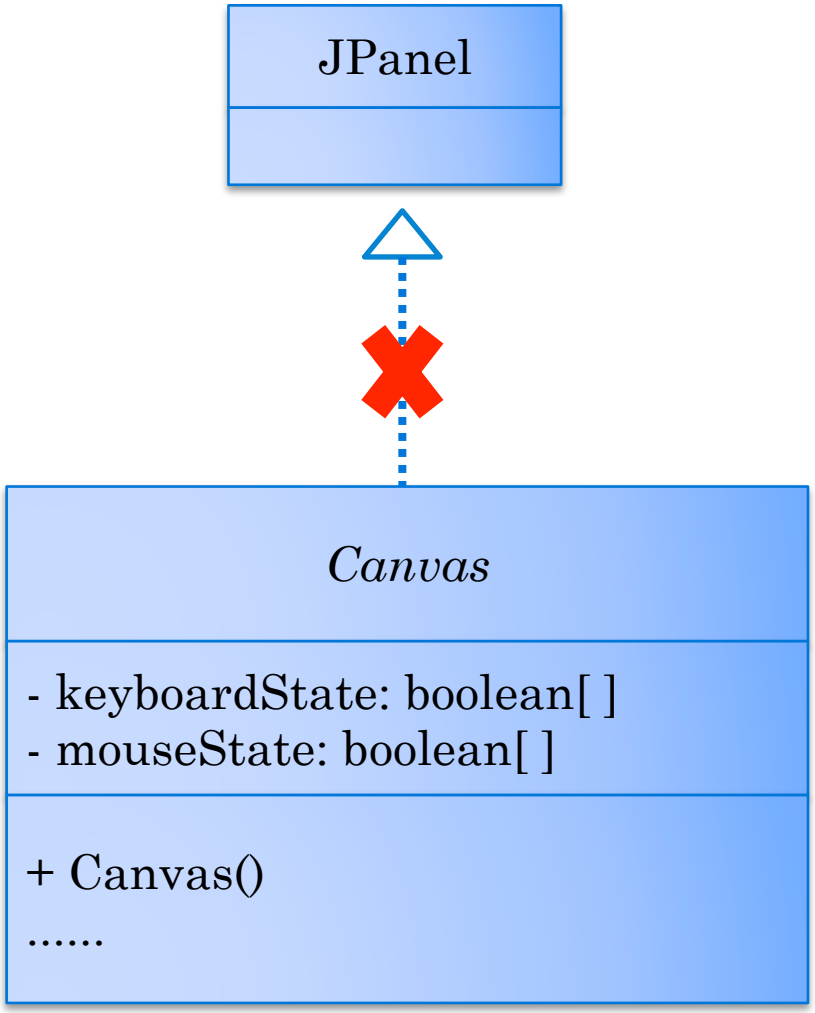
Vererbung:
Oberklasse ist eine Generalisierung der Unterklasse
„Is-a“ Beziehung

Vererbungspfeile nicht beschriften

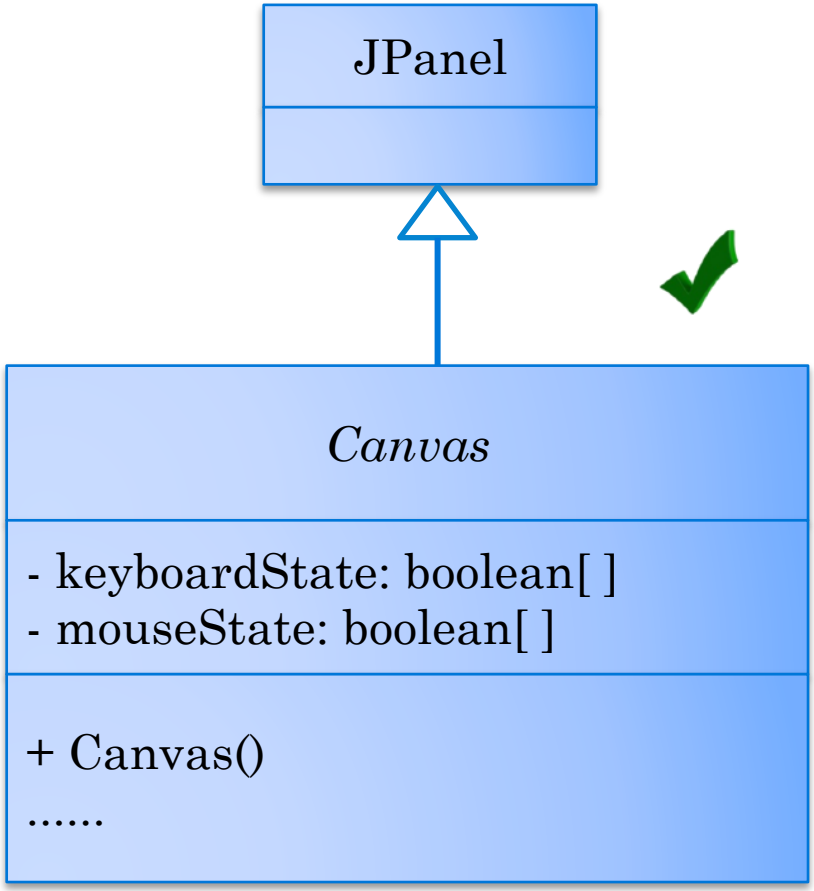
Kritik Vererbung



Kritik Vererbung



Kritik Vererbung

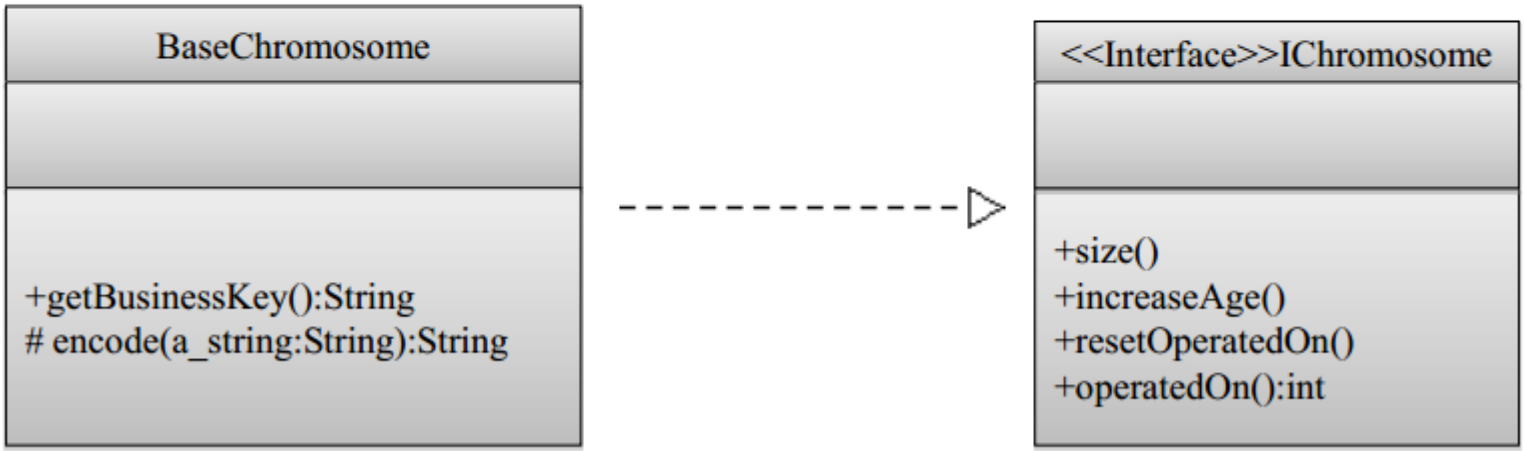


Kritik Interfaces

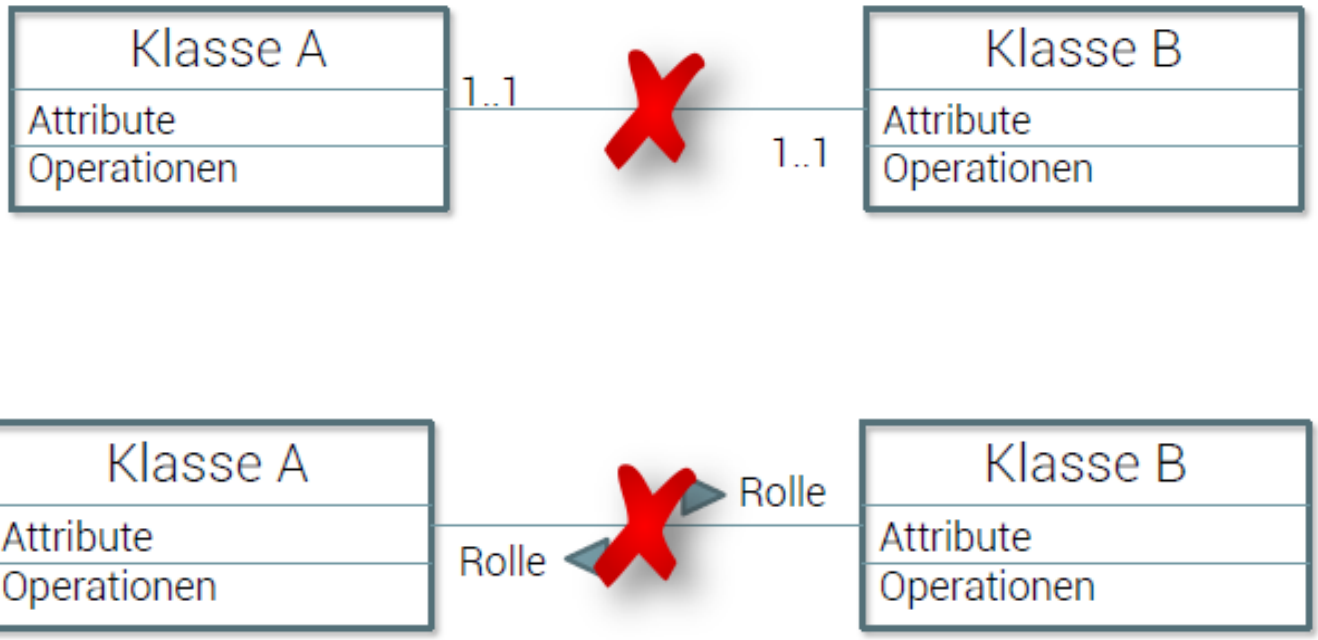
Realisierung:
Beziehung zu einer Schnittstelle

Kritik Interfaces

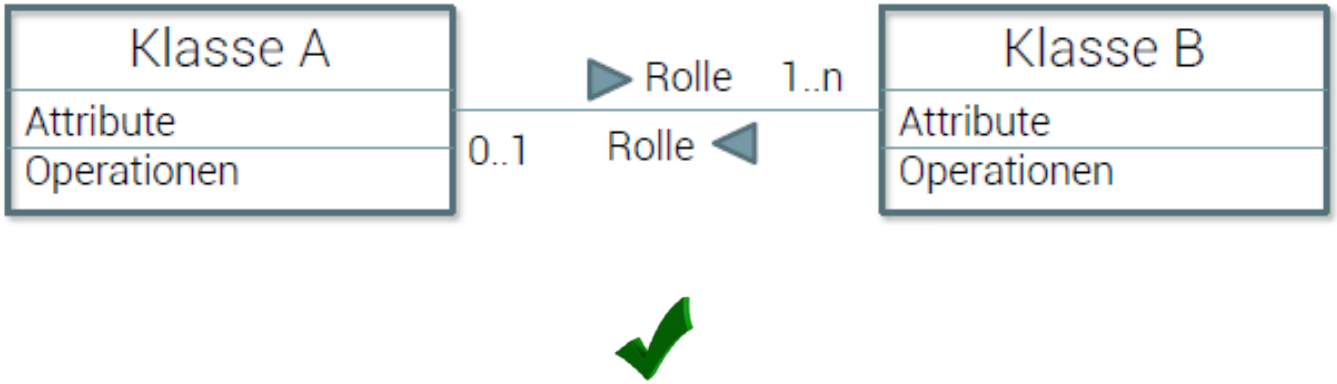
Realisierung:
Implementierte Methoden eines Interfaces werden nur im Interface modelliert



Kritik Assoziation

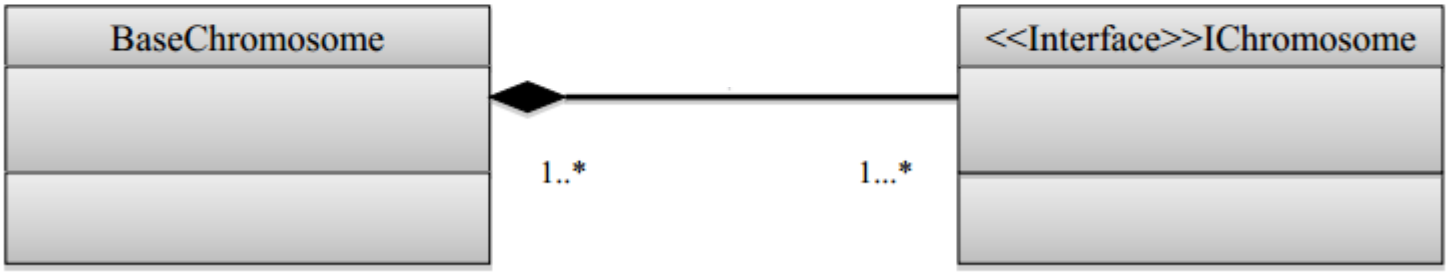


Kritik Assoziation



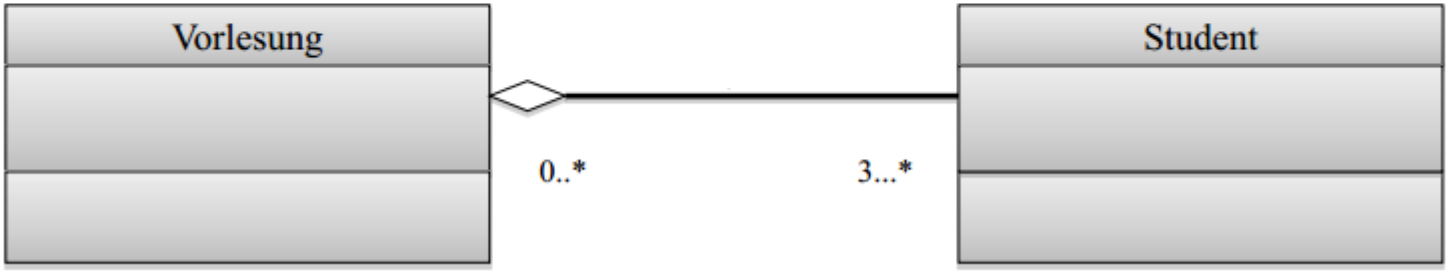
Kritik Assoziation

Komposition:
Teil kann nur existieren wenn Ganzes auch existiert



Kritik Assoziation

Aggregation:
Teil kann existieren auch wenn Ganzes nicht existiert



Aufgabe 3.2 Methoden: mouseKeyStatus

Code Verständnis:

```
private void mouseKeyStatus(MouseEvent e,
        boolean status)
{
    if(e.getButton() == MouseEvent.BUTTON1)
        mouseState[0] = status;
    else if(e.getButton() == MouseEvent.BUTTON2)
        mouseState[1] = status;
    else if(e.getButton() == MouseEvent.BUTTON3)
        mouseState[2] = status;
}
```

Diese Methode wird aufgeruft, wenn der Benutzer eine Maustaste drückt.

Diese Methode aktualisiert den boolischen Wert des als Parameter gegenbenden Buttons.

Aufgabe 3.2 Methoden: MouseButtonState

Code Verständnis:

```
private static boolean[] mouseState = new boolean[3];


public static boolean mouseButtonState(int
        button)
{
    return mouseState[button - 1];
}
```

Die Methode returniert den boolischen Wert «true» oder «false» für den als Parameter gegebenen Integer.

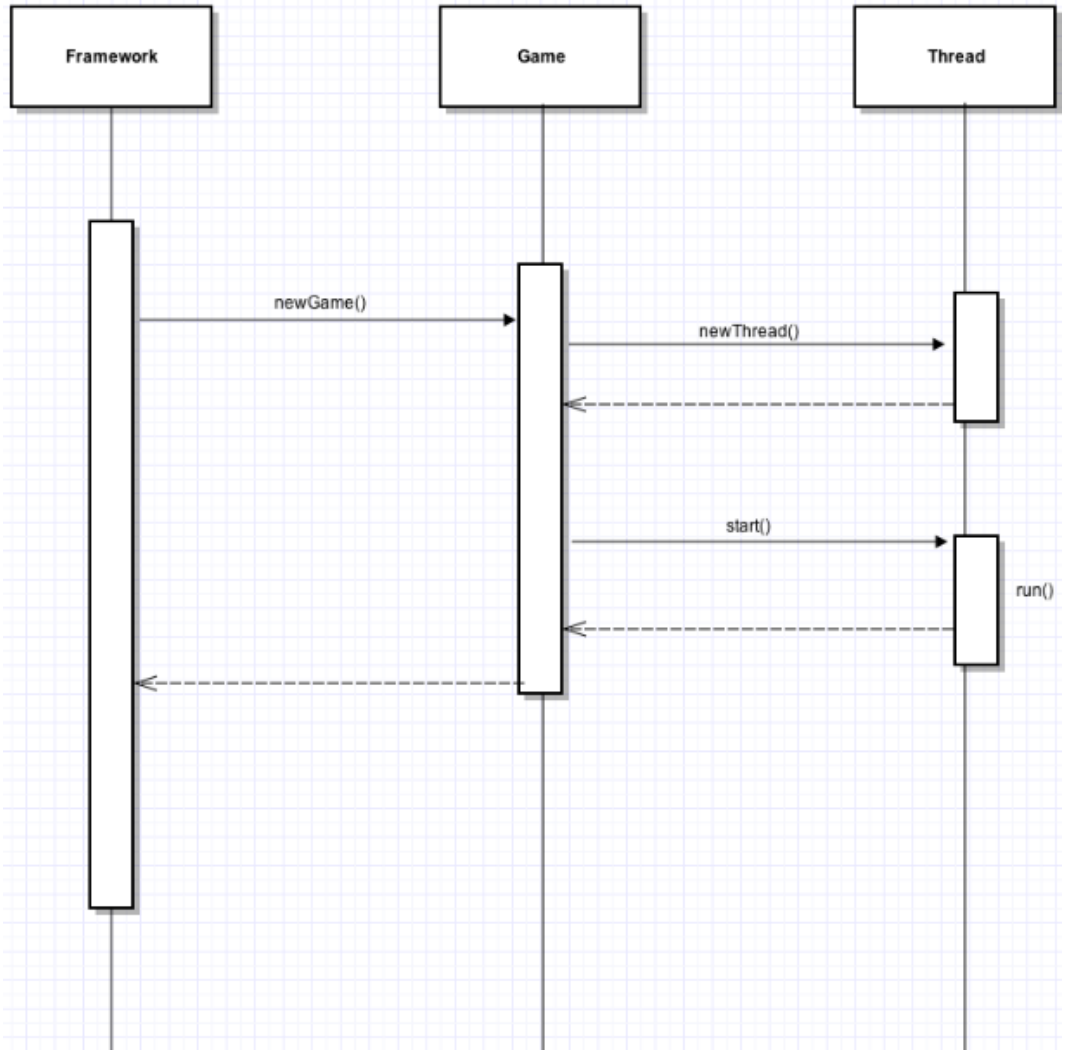
Die Methode prüft kontinuierlich, was die Maus tut.

Aufgabe 3.3 Verhalten: Linke Maustaste am Anfang

In der Klasse Framework ist die Methode «mouseClicked(MouseEvent e)» implementiert.

Wenn die linke Maustaste geklicked wird, wird ein MouseEvent der Methode mouseClicked übergeben. Diese initialisiert und startet das GUI des Spieles.

Aufgabe 3.3 Sequenzdiagramm

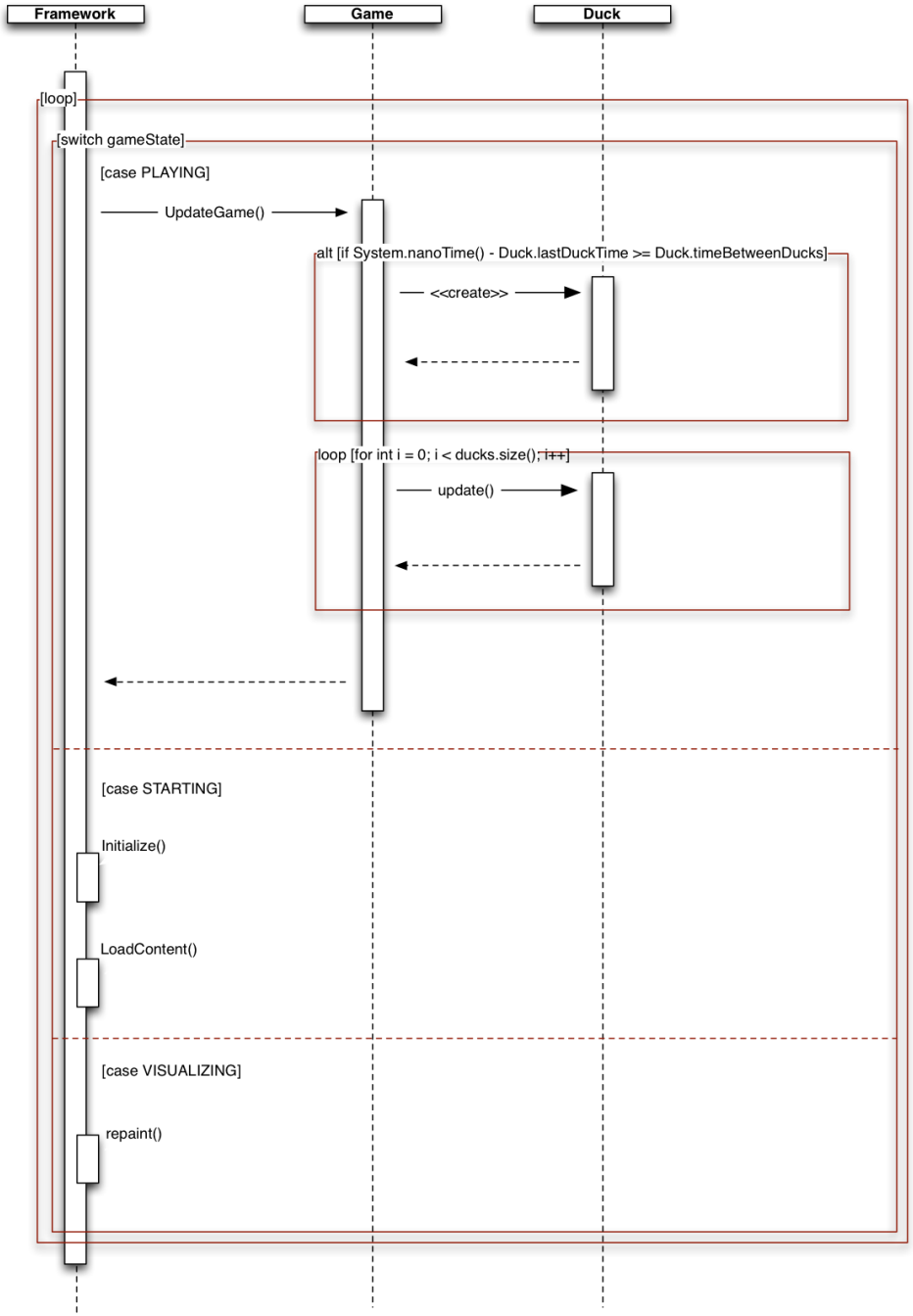


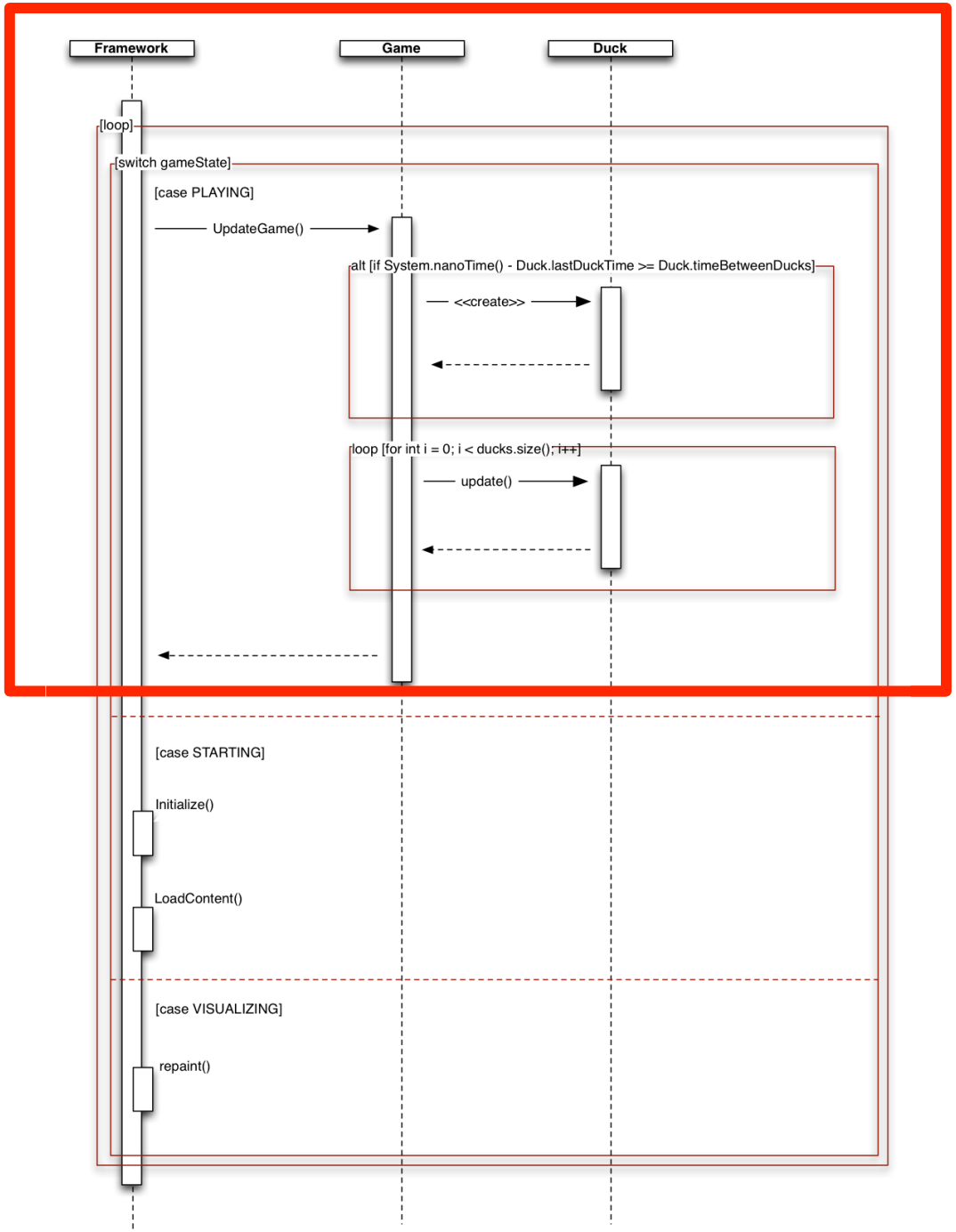
Aufgabe 3: Verhalten GameLoop

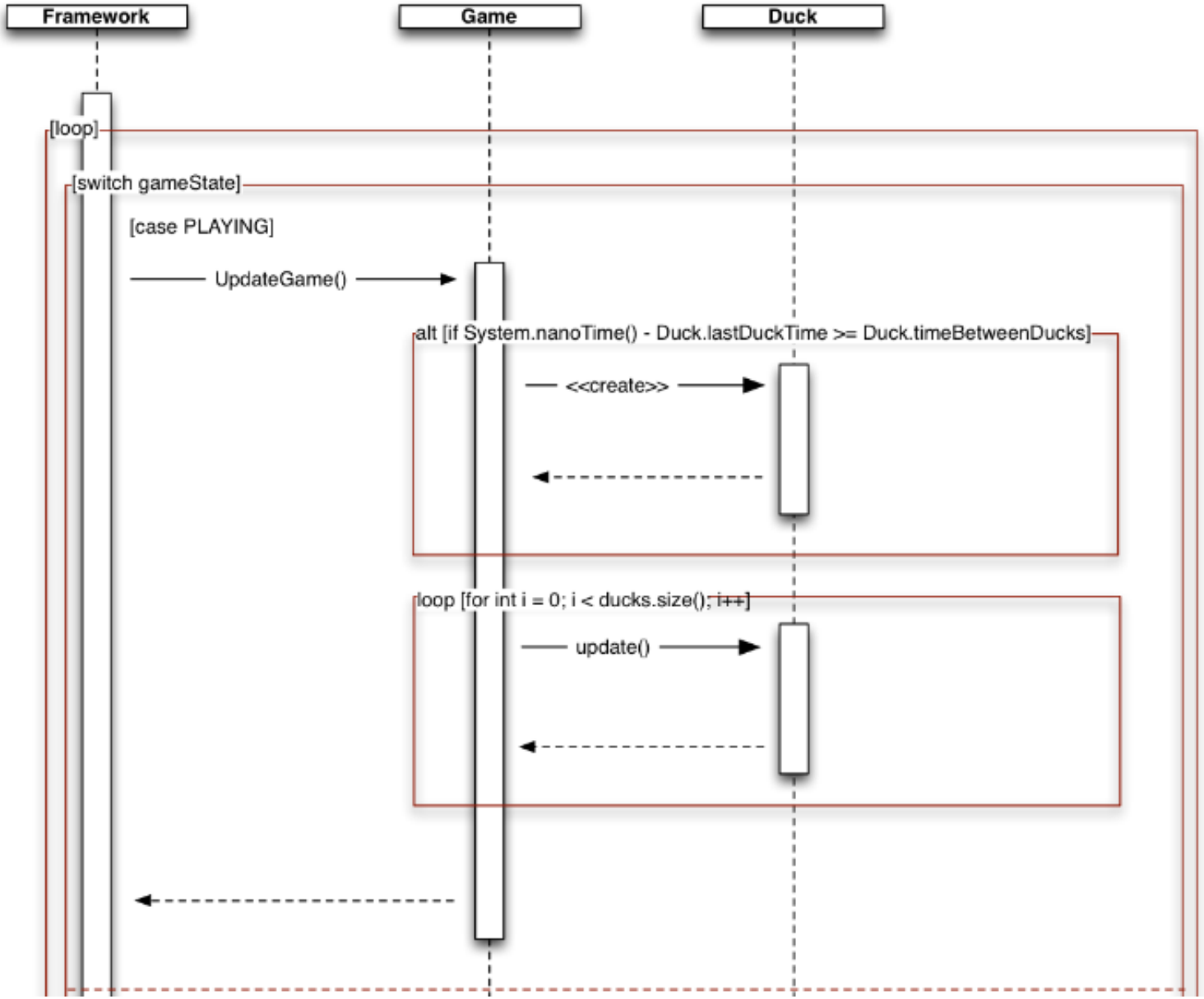
Switch Statement schaut in welchem Status sich das Spiel befindet, und aktualisiert den Screen dementsprechend.

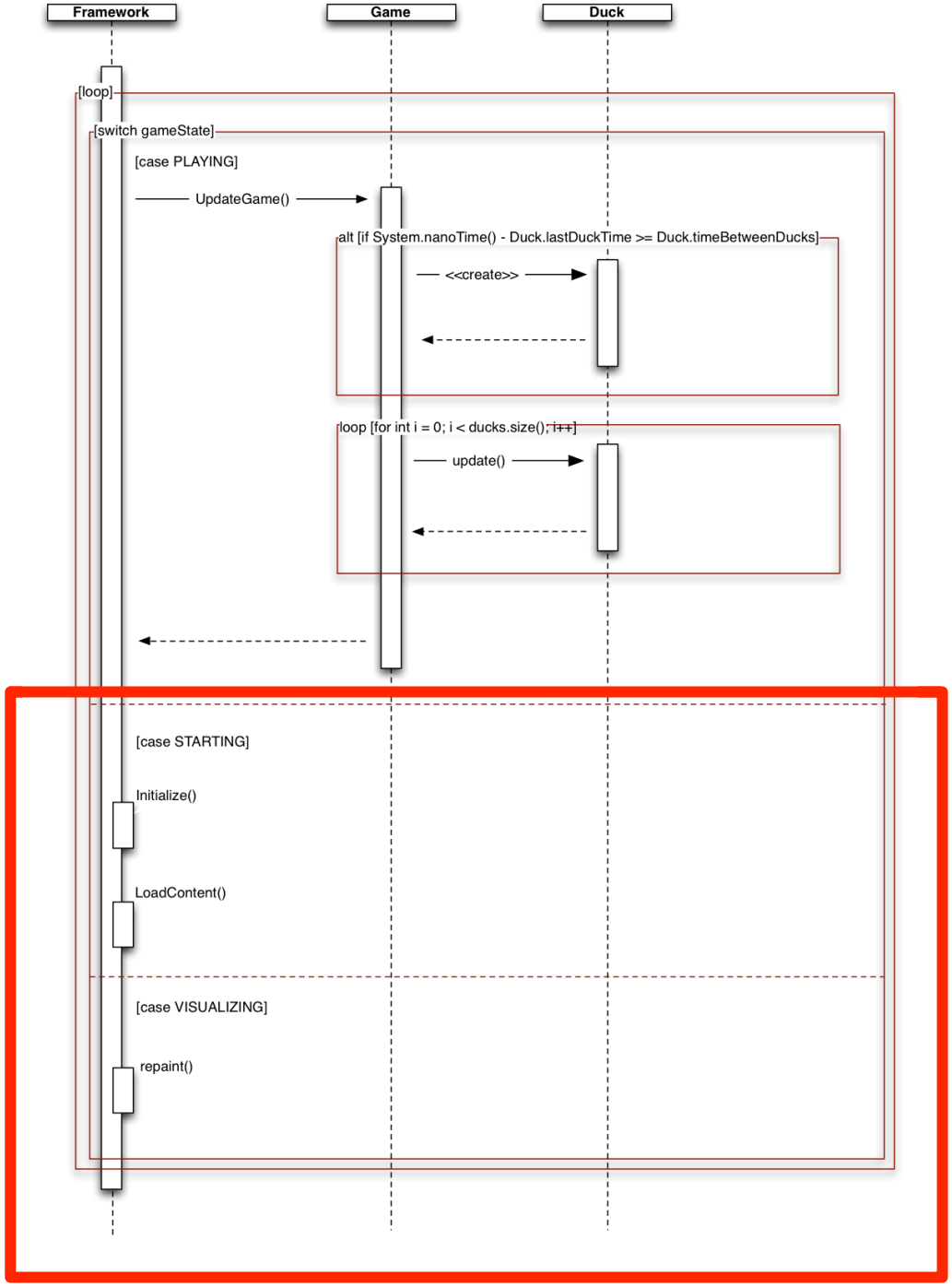
Sequenzdiagramm:

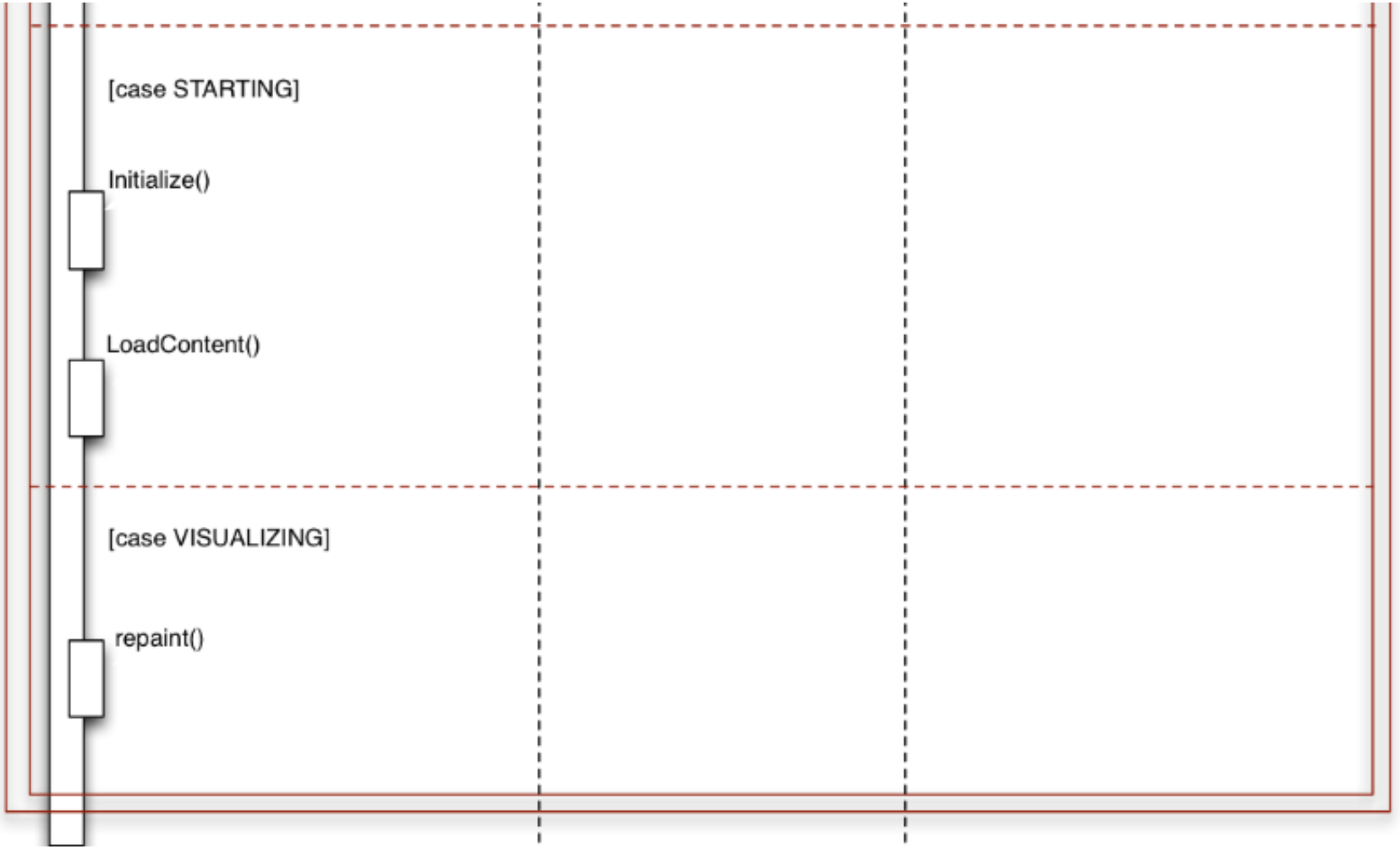
Siehe Übungsbesprechung und Korrekturen


Framework
Game
Duck
[loop]
[switch gameState]
[case PLAYING]
UpdateGame()
alt [if System.nanoTime() - Duck.lastDuckTime >= Duck.timeBetweenDucks]
<<create>>
loop [for int i = 0; i < ducks.size(); i++]
update()
[case STARTING]
Initialize()
LoadContent()
[case VISUALIZING]
repaint()


Framework
Game
Duck
[loop]
[switch gameState]
[case PLAYING]
UpdateGame()
alt [if System.nanoTime() - Duck.lastDuckTime >= Duck.timeBetweenDucks]
<<create>>
loop [for int i = 0; i < ducks.size(); i++]
update()
[case STARTING]
Initialize()
LoadContent()
[case VISUALIZING]
repaint()

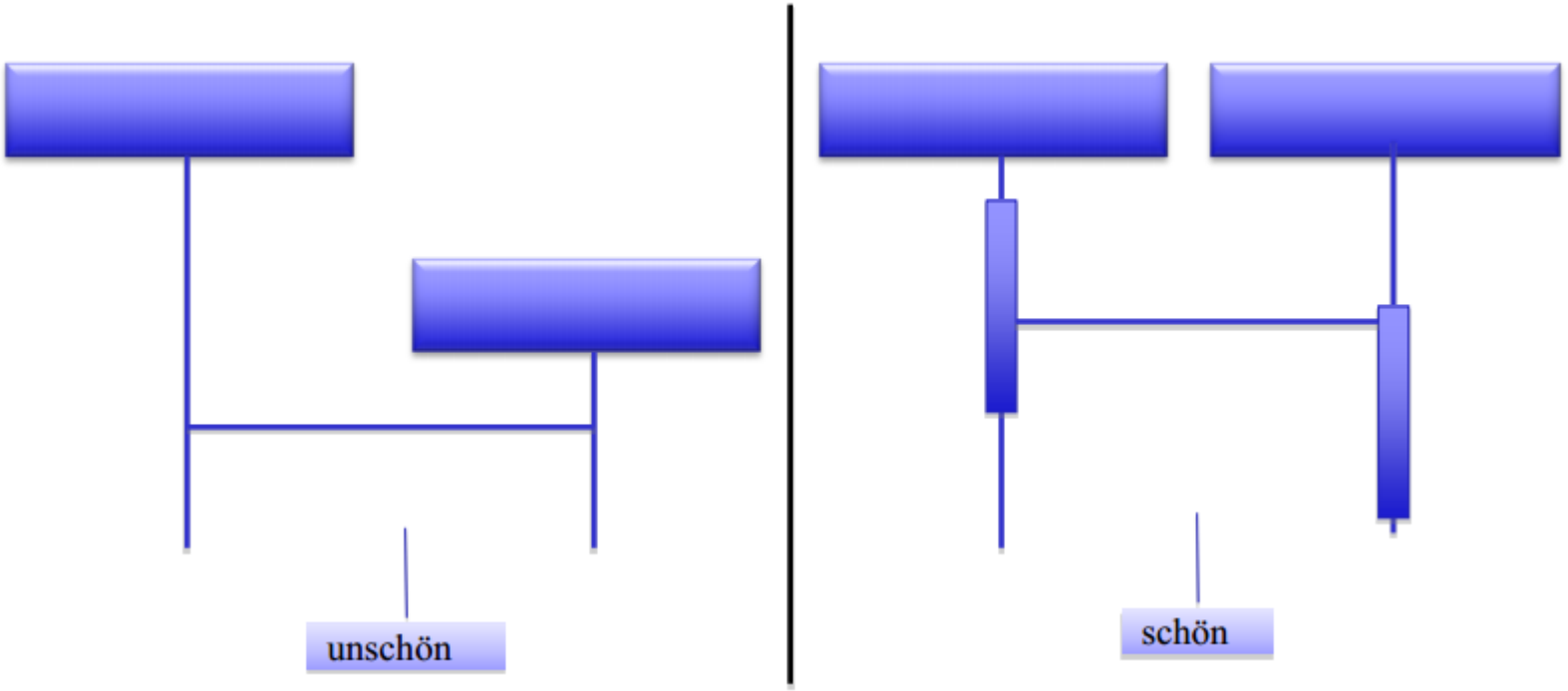

Framework
Game
Duck
[loop]
[switch gameState]
[case PLAYING]
UpdateGame()
alt [if System.nanoTime() - Duck.lastDuckTime >= Duck.timeBetweenDucks]
<<create>>
loop [for int i = 0; i < ducks.size(); i++]
update()


Framework
Game
Duck
[loop]
[switch gameState]
[case PLAYING]
UpdateGame()
alt [if System.nanoTime() - Duck.lastDuckTime >= Duck.timeBetweenDucks]
<<create>>
loop [for int i = 0; i < ducks.size(); i++]
update()
[case STARTING]
Initialize()
LoadContent()
[case VISUALIZING]
repaint()


[case STARTING]
Initialize()
LoadContent()
[case VISUALIZING]
repaint()

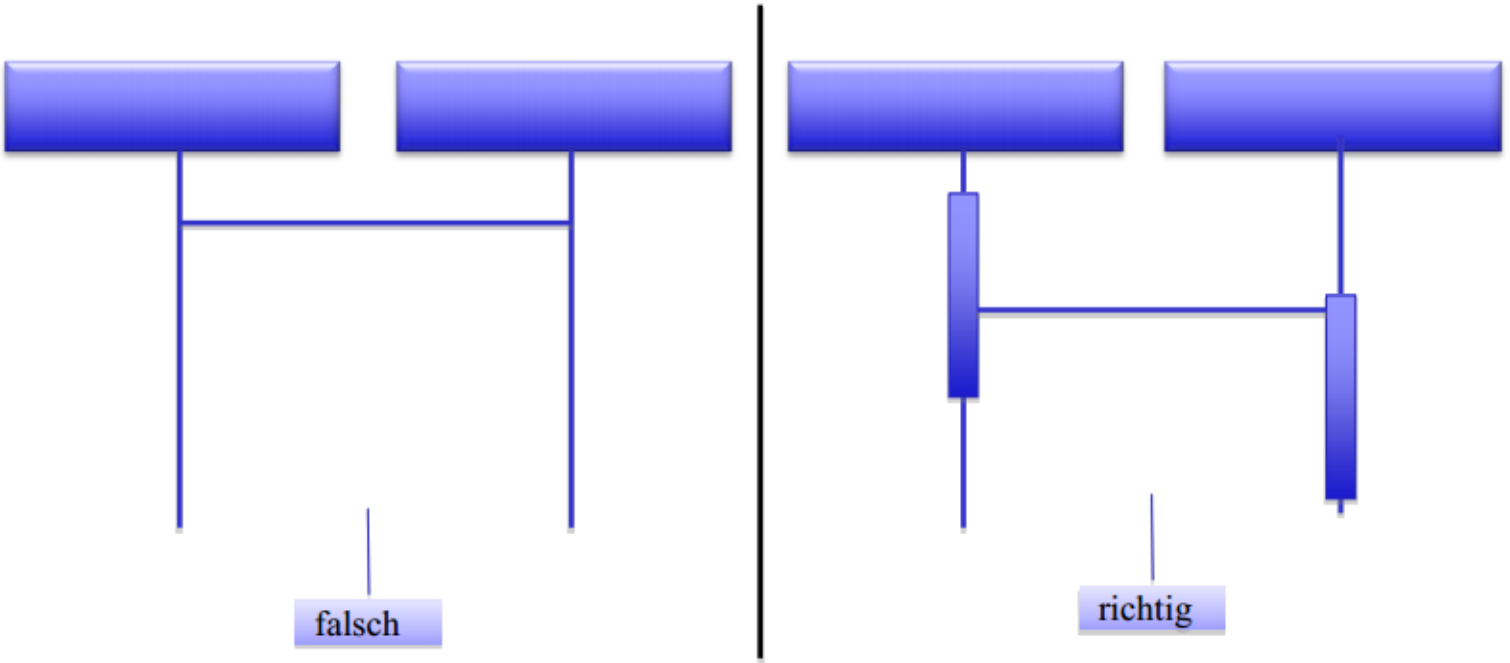
Kritik Sequenzdiagramm

- Auf die Darstellung achten:



Kritik Sequenzdiagramm

- Aktivierungsbereiche nicht eingezeichnet



Aufgabe 4: Verbessern Des Codes

4.1: Programmierstil

- Methodennamen schreibt man klein
  ➔ updateGame()

4.2: Dokummentation des Codes

Kritik:

- Nicht alle Kommentare ausgefüllt
- Nicht Code nacherzählen, sondern erklären warum etwas geschieht

Aufgabe 4: Verbessern Des Codes UpdateGame Methode

```
/**
 * Update game logic.
 *
 * @param gameTime gameTime of the game.
 * @param mousePosition current mouse position.
 */
public void UpdateGame(long gameTime, Point mousePosition)
{
    // Creates a new duck, if it's the time, and add it to the array list.
    if(System.nanoTime() - Duck.lastDuckTime >= Duck.timeBetweenDucks)
    {
        // Here we create new duck and add it to the array list.
        ducks.add(new Duck(Duck.duckLines[Duck.nextDuckLines][0] +
                random.nextInt(200), Duck.duckLines[Duck.nextDuckLines][1],
                Duck.duckLines[Duck.nextDuckLines][2],
                Duck.duckLines[Duck.nextDuckLines][3], duckImg));
```

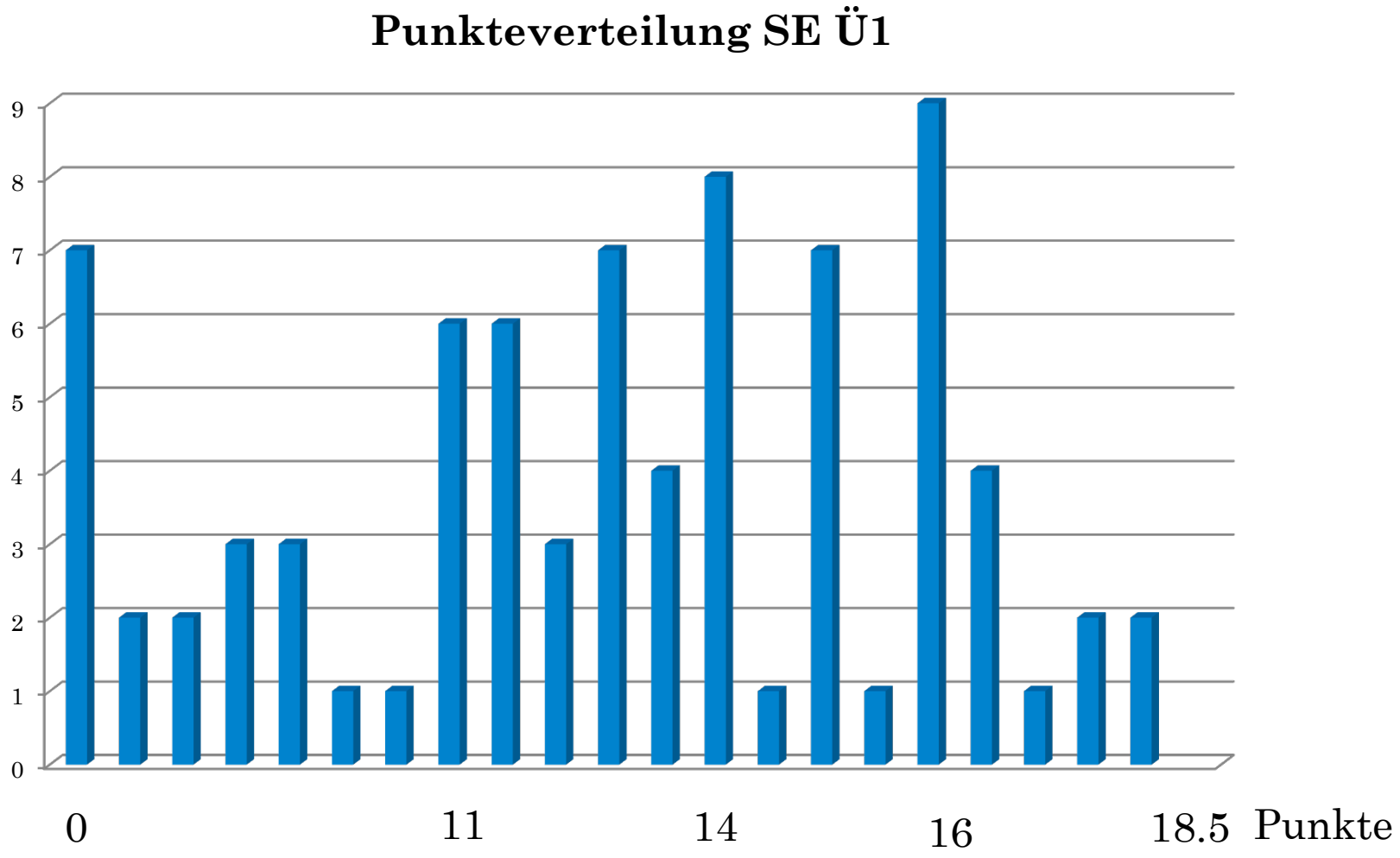
Aufgabe 4: Verbessern Des Codes

4.3: Änderungen im Code

- Implementation des Cancel-Knopfes

```
    //Der code fuer den Button
    JButton exitButton = new JButton("Spiel Beenden");
    exitButton.addActionListener (new SpielBeendenAction());
    this.add(exitButton);
    this.setVisible(true);
}
static class SpielBeendenAction implements ActionListener {
      public void actionPerformed (ActionEvent e) {
          System.exit(0);
      }
    }
```

PunkteVerteilung



Fragen?