**6. Menüs**

**Aufgaben:**

Implementieren Sie den Menüpart eines Programms. Dabei soll es ein Menü namens „Player“ geben. Es sollen Spieler über das Item „add Player“ hinzugefügt und über „remove Player“entfernt werden. Mit *JoptionPane.showInputDialog(Component parent, String message)* können sie einen Dialog öffnen, damit der User jeweils den Namen des Spielers angibt. Die Methode *showInputDialog* gibt einen String zurückgibt.

```
public MyPlayerMenu() {

	ArrayList<Player> player = new ArrayList<>();
	JMenu menu = new JMenu("Player");
	JMenuItem addPlayer = new JMenuItem("add Player");
	JMenuItem removePlayer = new JMenuItem("Remove Player");

	setJMenuBar(new JMenuBar());
	getJMenuBar().add(menu);
	menu.add(addPlayer);
	menu.add(removePlayer);

	addPlayer.addActionListener(listener -> {
		String name = JOptionPane.showInputDialog(this, "add Player\nName:");
		player.add(new Player(name));
	});

	removePlayer.addActionListener(listener -> {
		String name = JOptionPane.showInputDialog(this, "remove player: \nName:");

		for (Player user : player) {
			if (user.getName().equals(name)) {
				player.remove(user);
				break;
			}
		}
	});


	setSize(300, 300);
	setDefaultCloseOperation(EXIT_ON_CLOSE);
	setVisible(true);
}

public class Player{

	private String name;

	public Player (String name) {
		this.name = name;
	}

	public String getName() {
		return name;
	}
}
```

**7. Graphics und Timer**

**Aufgabe:**

Ein Rechteck soll sich nach links bewegen.
Erzeuge einen Timer und änder entsprechend die x-Koordinate des Rechtecks.
Was darfst du auf keinen Fall vergessen, damit man die Änderung auch sehen kann?

```
private Rectangle rect;

public MovingRect() {

	rect = new Rectangle(5, 5, 100, 100);
	MyPanel panel = new MyPanel();

	add (panel);

	Timer t = new Timer(20, listener -> {
		rect.x++;

		panel.repaint();
	});

	t.start();


	setSize(300, 300);
	setDefaultCloseOperation(EXIT_ON_CLOSE);
	setVisible(true);
}

private class MyPanel extends JPanel{

	@Override
	protected void paintComponent(Graphics g) {
		super.paintComponent(g);

		g.setColor(Color.BLUE);
		g.drawRect(rect.x, rect.y, rect.width, rect.height);

	}
}
```

**8. Threads**

**Aufgabe:**

Ihr seid als Backendprogrammierer verantwortlich, dass das Buchungssytem eines Fluganbieters einwandfrei funktioniert. Dabei gibt eine Klasse Flugbuchung, die eine ArrayListe von Flügen hat.
Ein Flug hat u.a. ein Datum (datum), eine Startort(startFH), einen Ankunftsort(zielFH). Schreiben Sie in der Klasse Flug eine Methode equals, die diese Attribute überprüft.
In der Klasse Flugbuchung soll eine Methode buchen sein, die bei dem Flug Objekt die Methode reservieren aufruft und so einen Platz reserviert. Sorgen Sie dafür, dass nur ein Benutzer zur Zeit auf die buchen()-Methode zugreifen kann.

```
public class Flug {

	private Date zeit;
	private String startFH;
	private String zielFH;
	private int freieSitze;

	//hier gibt es noch weitere Methoden wie Getter und  einen Konstruktor

	public boolean reservieren (int sitze) {
		if (freieSitze >= sitze) {
			freieSitze -= sitze;
			return true;
		}
		return false;
	}

	@Override
	public boolean equals(Flug other) {

		if (!startFH.equals(other.startFH)) {
			return false;
		}

		if (!zeit.equals(other.zeit)) {
			return false;
		}

		if (!zielFH.equals(other.zielFH)) {
			return false;
		}
		return true;
	}
}


public class Flugbuchung {

	private ArrayList<Flug> fluege;

	public Flugbuchung() {
		fluege = new ArrayList<>();

		//hier wird noch die ArrayListe befüllt
	}

	//hier gibt es noch weitere Methoden
```

```
	public synchronized boolean buchen (Flug flug, int anzahlSitze) {

		if (fluege.contains(flug)) {

			int index = fluege.indexOf(flug);
			return fluege.get(index).reservieren(anzahlSitze);

		}
		return false;
	}
}
```

**Fragen:**

**9. Sockets**

**Aufgabe:**
Ein User soll wie bei Whatsapp eine Nachricht an einen anderen User schicken können. Dabei wird als erstes die ID (String) des Absenders, dann die ID des Empfängers und dann die Nachricht versendet. Der Server hat eine Hashmap <String, Socket> users, in denen jeder ID ein Socket zugeordnet sein kann.
Schreibe die Methode receive, die eine Nachricht empfängt und sie an den entsprechenden User weiterleitet.

```
public class Server {

	private HashMap<String, Socket> user = new HashMap<>();

	public Server() {
		//hier wird die Verbindung zu dem User aufgebaut etc.
	}

	public void empfangen(Socket socket) {
		String from = "";
		String to = "";
		String message = "";
		PrintWriter writeToOther;
		Scanner reader = socket.getInputStream();

		while (isRunning) {
			try {
				from = reader.nextLine();
				to = reader.nextLine();
				message = reader.nextLine();

				writeToOther = new
						PrintWriter(user.get(to)).getOutputStream());
				writeToOther.println(message);
				writeToOther.flush();
				writeToOther.close();

			} catch (Exception e) {
			}
		}
	}
}
```

**Fragen:**

Was ist ein Serversocket? Was kann der Server damit machen? Ein ServerSocket hat zwei Parameter, wofür stehen diese?

Ein Server erstellt einen Serversocket, um über einen angegebenen Port auf eine Verbindungsanfrage eines Clients zu warten. Ein Serversocket hat die Parameter (int) port und die Anzahl der möglichen Verbindungen (int).

Über welche Schnittstelle kommunizieren Client und Server mit einander?

Über Sockets, ein Client baut ein Socket direkt auf, wenn er eine Verbindungsanfrage zu dem Server aufbaut. Der Server bekommt das Socket als Rückgabewert der accept-Methode, wenn ein sich Client mit dem Server verbunden hat.