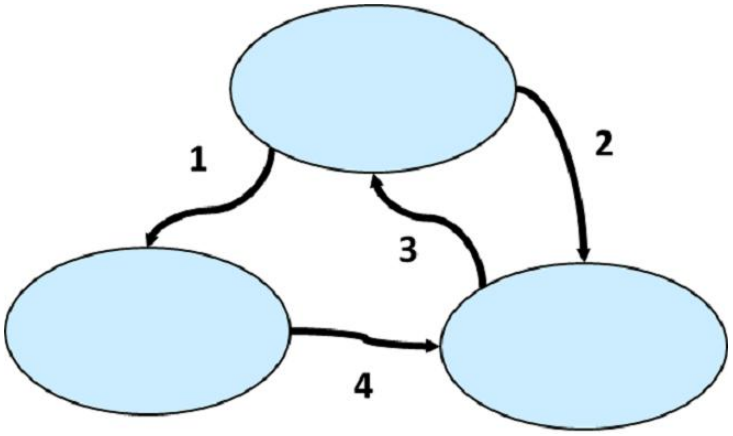
Hochschule für Technik und Wirtschaft Berlin
University of Applied Sciences

Aufgabe 1
Das Betriebssystem hat unter anderem die Aufgabe Prozesse zu verwalten.
Prozesse können aber verschiedene Zustände besitzen:

- Nennen Sie die 3 Zustände, die ein Prozess haben kann und tragen Sie dies in die vorliegende Skizze ein!

- Vier Übergänge sind zwischen diesen Zuständen möglich. Erläutern Sie diese Zustandsübergänge!



Aufgabe 2
Unter anderem sorgt das Betriebssystem für die Anbindung benötigte Geräte:
- Was verstehen Sie unter Gerätemanager?
- Welche Aufgaben hat die Geräteverwaltung?

Aufgabe 3
Auf einer Rechenanlage werden viele Daten dauerhaft gespeichert und müssen entsprechend verwaltet werden:
- Was macht die Dateiverwaltung?
- Welche Operationen muß ein Betriebssystem sicherstellen?

Aufgabe 4
Öffnen Sie IrfanView. Drücken Sie nun die Tastenkombination STRG+ALT+ENTF und öffnen Sie den Task-Manager. Suchen Sie unter Prozesse „IrfanView“.
- Welche Prozesskennung hat die Instanz ihres Programms?
- Weshalb wird eine Prozesskennung benötigt?
- Führen Sie einen Rechtsklick aus und wählen sie „Task beenden“.
- Scenario: Ihr Windows Explorer hat sich aufgehängt. Wie gehen Sie vor?

Aufgabe 5
Das Betriebssystem steuert und überwacht den geregelten Ablauf der Prozesse auf dem Rechner:
a) Definieren Sie den Begriff „Prozess“ in diesem Kontext!
b) Was ist der Unterschied zwischen „Prozess“ und „Programm“?

Aufgabe 6
Der Scheduler ist eine wichtige Komponente des Betriebssystems, die für einen geregelten Ablauf der Prozesse sorgt:
a) Was verstehen Sie unter Scheduler?
b) Welche Kriterien werden beim Scheduling-Verfahren berücksichtigt?
c) Welche Scheduling Strategien kennen Sie? Nennen Sie min. 3 Verfahren und erläutern Sie diese kurz!

Aufgabe 7
Nennen Sie den Unterschied zwischen einem Compiler und einem Interpreter

Aufgabe 8
Erstellen Sie mit dem Windows Aufgabenplaner ein automatisches Backup Ihrer wichtigen Dateien zu einem regelmäßigen Zeitpunkt. Nutzen Sie hierzu ein hinterlegtes Script.

Hochschule für Technik und Wirtschaft Berlin
University of Applied Sciences

Aus Alt Mach Neu

*Softwareentwickler*innen kommen oft in die Situation, dass sie Programme verbessern müssen, die alt sind, die irgendein*e Praktikant*in mal geschrieben hat oder die sie selber geschrieben haben, als sie noch nicht so viel Erfahrung hatten. Ein Programm nehmen und es verbessern heißt „Refactoring". Dabei wird der Code eleganter gemacht, die Architektur der Software kann verbessert werden, das Programm kann auch ressourcenschonender gemacht werden.*

In der Aufgabe von Tag 3 hast du schon ein bisschen Refactoring gemacht, nämlich als du doppelten Code durch eine Funktion mit Parametern ersetzt hast.

In dieser Aufgabe übst du, wie du gegebenen Code eleganter machst. Dabei übst du Programmatic Thinking und fremden Code nachzuvollziehen.

1. **Remixe das Projekt „Bananenspiel"** (Original von Flox1210): https://scratch.mit.edu/projects/220090548/

2. **Spiele das Spiel!**

3. **Untersuche den Code**: Welche Figuren gibt es? Was machen diese? Welche Events lösen welchen Code aus?

Tipp: Mach dir Notizen bzw. male ein Diagramm

Tipp: Welche verschiedenen Schwierigkeitsstufen soll es geben, und wie unterscheiden sich diese?

4. **Recherchiere im Scratch-Wiki unter** https://de.scratch-wiki.info/wiki/Klonen**, wie die Klonen-Funktion bei Scratch funktioniert.**

5. **Recherchiere im Scratch-Wiki unter** https://de.scratch-wiki.info/wiki/Senden**, wie die Senden-Funktion bei Scratch funktioniert.**

6. **Mach dir einen Plan, was du alles refactoren könntest, um den Code möglichst elegant zu machen.** Also: Wie kannst du doppelten Code vermeiden? Wie kannst du Parameter und Variablen, Conditionals, Klone und Funktionen geschickt nutzen?

Tipp: Könnte man das Management der Schwierigkeitsstufen vereinfachen? Z.B. mit einer Variable?

Tipp: Schachtle Schleifen und Conditionals geschickt, um doppelten Code zu vermeiden. Wann ist doppelter Code für die Funktionalität des Programmes notwendig?

Tipp: Gibt es Code der nie ausgeführt wird?

7. **Refactore dein Spiel, Schritt-für-Schritt**. Z.B. ersetze erst doppelten Code in einer Figur durch eine Funktion.

Tipp: Teste nach jedem Schritt, ob dein Programm funktioniert.

8. **Füge dein fertiges Spiel zur Gruppe unter** https://scratch.mit.edu/studios/27659504/ **hinzu.**

9. **Schau dir refactored Spiele der anderen an und reflektiere.** Was haben sie anders gemacht als du? Was denkst du, ist die elegantere Lösung? Haben die anderen Lösungen auch Vorteile?

10. **Refactore dein Spiel erneut.**

11. Optional: Individualisiere das Spiel, wie es dir gefällt und verbessere die Funktionalität!

Google Engine effektiv nutzen:
https://www.lifehack.org/articles/technology/20-tips-use-google-search-efficiently.html
HTW-Hochschulsport:
https://hochschulsport.htw-berlin.de/
Studierendenschaft der HTW:
https://students-htw.de/