A1 Extreme Programming

Nennen Sie mögliche Gründe, warum Sustainable Pace, also eine angemessene Arbeitsbelastung und nachhaltige Geschwindigkeit bei den Praktiken von Extreme Programming gefordert wird.

Lösung 1

A2 Test-Driven Development

Sie möchten für Ihr Smart Home eine Zeitsteuerung für eine Lampe entwickeln. Formulieren Sie umgangssprachlich fünf Testfälle, die Ihr zu entwickelndes Softwaresystem erfüllen sollte.

Lösung 2

A3 Scrum

a) Erläutern Sie, wie man Ihrer Meinung nach bei Scrum vorgehen sollte, wenn das Entwicklungsteam im Laufe eines Sprints folgendes erkennt:

   i. Das Sprintziel und die User Stories im Sprint Backlog können vorzeitig vor Ende des Sprints erfüllt werden.

   ii. Das Sprintziel und die User Stories im Sprint Backlog können sehr wahrscheinlich nicht rechtzeitig zum Ende des Sprints erfüllt werden.

b) Nennen Sie die Vor- und Nachteile einer sehr kurzen Sprintlänge (z.B. eine Woche) gegenüber einer sehr langen Sprintlänge (z.B. acht Wochen).

Lösung 3

A4 Smartphone

Modellieren Sie die folgende Steuerung für ein Smartphone als UML-Zustandsdiagramm:

a) Das Smartphone kann sich in den Zuständen Ausgeschaltet, Standby, Energiesparen und Normal befinden.

b) Ereignisse für Zustandsübergänge sind das Drücken der An/Aus-Taste, der Home-Taste oder eine Bildschirmgeste. Diese Ereignisse sollen als Signale modelliert werden.

c) Nach 15 Sekunden Inaktivität im Normalzustand wird der Energiesparmodus aktiviert. Dieser kann durch Drücken der Home-Taste beendet werden.

C.R. Thelen, L. Hellwig, S. Menne

d) Nach 60 Sekunden im Energiesparmodus wird der Standby-Betrieb aktiviert. Durch Drücken der Home-Taste gelangt man wieder in den Normalbetrieb.

e) Das Smartphone kann im Normalbetrieb über die An/Aus-Taste ausgeschaltet werden. Ist das Smartphone ausgeschaltet, dann kann es über die An/Aus-Taste eingeschaltet werden und man gelangt in den Normalbetrieb.

Lösung 4

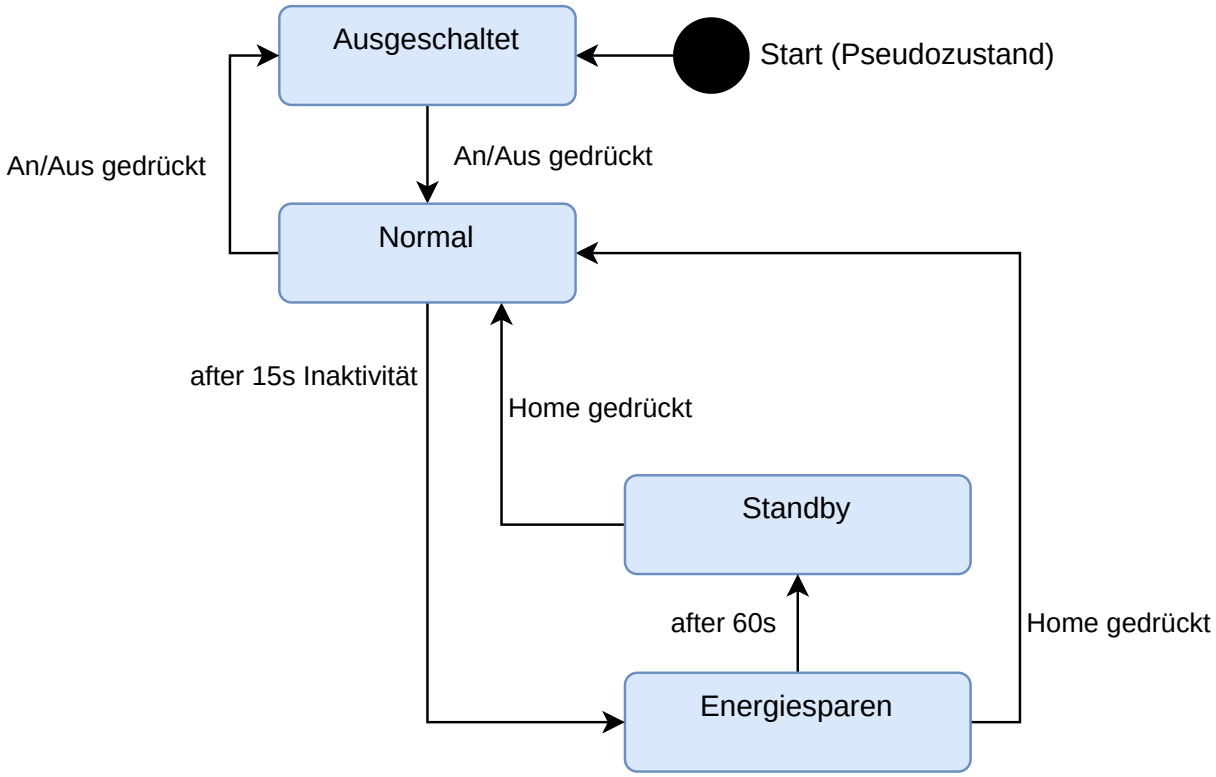

Abbildung 1: Euklidischer Algorithmus

A5 Stapelspeicher

Sie wollen eine parametrisierte Klasse Stack für einen Stapelspeicher mit generischem Datentyp `TElement` für die Elemente implementieren. Für den Klassenparameter sollen alle Referenztypen erlaubt sein. Die Klasse soll die schreibgeschützten Attribute Capacity für die maximale Anzahl der Elemente und Count für die tatsächliche Anzahl der Elemente besitzen, beides vom Datentyp Integer. Die Kapazität soll über den Konstruktor gesetzt werden können und ist immer mindestens 2.
Die möglichen Zustände des Stacks sind:

- Empty: der Stapelspeicher enthält keine Elemente

- Filled: der Stapelspeicher enthält Elemente und kann noch neue Elemente aufnehmen
- Full: der Stapelspeicher ist voll und kann keine neuen Elemente mehr aufnehmen

Die Klasse hat die folgenden öffentlichen Operationen:

- Push: lege das übergebene Element (vom Datentyp `TElement`) auf dem Stapelspeicher ab, wenn dieser noch nicht voll ist; ansonsten passiert nichts
- Pop: wenn der Stapelspeicher Elemente enthält, dann liefere das oberste Element zurück und entferne es vom Stapel; ansonsten wird NULL zurückgeliefert
- Peek: wenn der Stapelspeicher Elemente enthält, dann liefere das oberste Element zurück, ohne es vom Stapel zu nehmen; ansonsten wird NULL zurückgeliefert

Modellieren Sie die Klasse Stack als UML-Klassendiagramm und alle Zustandsübergänge als UML-Zustandsdiagramm. Nutzen Sie bei den Transitionen Call Events, Guards und Effects.

Lösung 5