Ahoi!

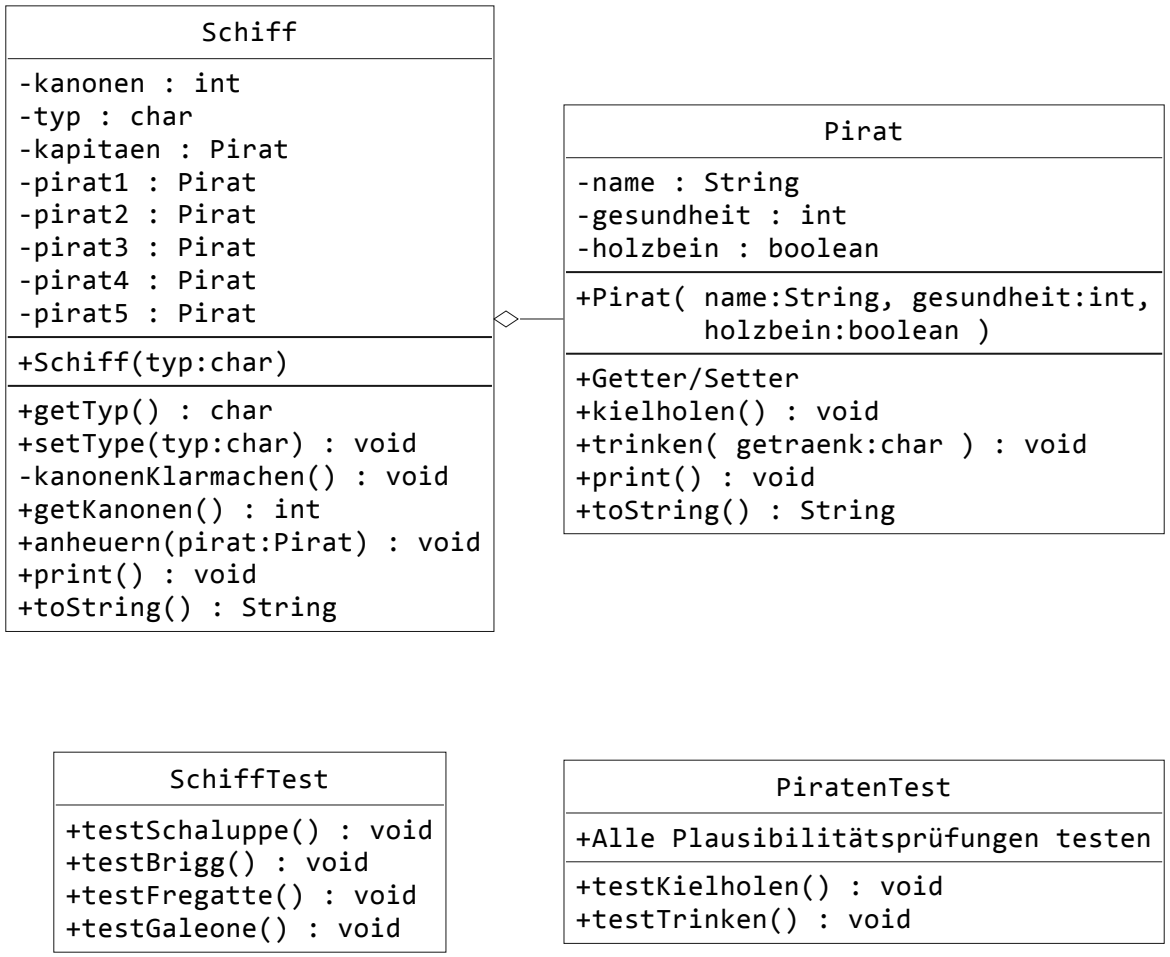

Abbildung 1. UML-Diagramm

Beschreibung

In diesem Beispiel sollen Basisklassen für ein Computerspiel implementiert werden, die später als Modellklassen im Programm dienen und mit Daten gefüllt werden. Für eine einwandfreie Funktion soll pro Klasse jeweils eine Testklasse erstellt werden, die sämtliche Methoden der Klasse testet.

- Führen Sie ausschließlich geforderte Prüfungen durch
- Halten Sie sich strikt an die Angabe
- Testen Sie Ihren Code ausführlich

Klasse Pirat

Implementieren Sie die Klasse gemäß des UML-Diagrammes. In der nachstehenden Tabellen finden Sie Details zu Prüfungen und Methoden.

Attribute

Tabelle 1. Beschreibung der Klasse Pirat

Attribute	Beschreibung
`name`	Name des Piraten
`gesundheit`	Bildet die Gesundheit in Prozent ab. (0 - 100 Prozent)
`holzbein`	Boolean Wert, der angibt, ob der Pirat ein Holzbein hat, oder nicht.

Methoden

Sämtliche Get- und Set- Methoden sind wie üblich zu implementieren. Die restlichen Methoden sowie durchzuführende Plausibilitätsprüfungen sind in der nachstehenden Tabelle beschrieben.

Methoden	Beschreibung
`setName(name)`	• Der Name darf nicht `null` sein • Wird ein unerlaubter Wert übergeben, so soll folgende Meldung auf die Konsole ausgegeben werden: `Arrgh! Keinen Namen fuer den alten Seebaeren uebergeben!`
`setGesundheit(gesundheit)`	• Der Wert Gesundheit ist ein Prozentwert. • Erlaubter Wertebereich: **0** bis **100** • Wird ein unerlaubter Wert übergeben, so soll folgende Meldung auf die Konsole ausgegeben werden: `Yo-ho-ho - Gültiger Bereich für die Gesundheit: 0 bis 100 Prozent`

Methoden	Beschreibung
`trinken(getraenk)`	**Erlaubte Werte:** • `w` - Wasser: Vermindert die Gesundheit um **10** Prozent • `g` - Grog: Erhöht die Gesundheit um **10** Prozent • `r` - Rum: Erhöht die Gesundheit um **20** Prozent **Beachten Sie:** • Die Gesundheit darf **nie** unter **0** Prozent sinken. • Die Gesundheit darf **nie** über **100** Prozent steigen. • Ist die Gesundheit auf **0** Prozent gesunken, so soll eine Meldung ausgegeben werden: `Haudegen NAME ging ueber die Planken.` • Wenn die Gesundheit wieder **100** Prozent erreicht hat, so soll eine Meldung ausgegeben werden: `Haudegen NAME erfreut sich voller Gesundheit`
`kielholen()`	• Ein Pirat erleidet beim Kielholen einen Schaden von **90** Prozent • Hat der Pirat ein Holzbein, so erleidet er beim Kielholen einen Schaden von **95** Prozent • Der Schaden soll von der Gesundheit des Piraten abgezogen werden • Die Gesundheit darf **nie** unter **0** sinken! • Hat die Gesundheit **0** Prozent erreicht, so soll folgende Meldung auf die Konsole ausgegeben werden: `Pirat NAME stieg in Davy Jone’s Kiste`

Methoden	Beschreibung

`toString()`
String
	• Liefert einen `String` im nachstehenden Format zurück (Returntyp)
	
	`` `Aye` - Trunkenbold NAME meldet sich an Board! GESUNDHEITS_INFO %, HOLZBEIN_INFO ``
	
	Beachten Sie folgende Anforderungen:
	
	• Setzen Sie für den `NAME` den Namen des Piraten ein
	• Folgende Anforderungen sind bei der `GESUNDHEITS_INFO` einzuhalten:
	  • Gesundheit von 11 bis 100 Prozent: Ausgabe der Gesundheit als Zahl
	  • Gesundheit von 1 bis 10 Prozent: Gesundheit als Zahl und Text : `- hisst den Yellow Jack`
	  • Gesundheit von 0 Prozent: `X - sprang in die Kiste`
	
	Beispiel:
	
	`` `Aye` - Trunkenbold Jack Sparrow meldet sich an Board! 100%, Zweifüßer `Aye` - Trunkenbold Barbarossa meldet sich an Board! 100%, Holzbeiner `Aye` - Trunkenbold Jack Sparrow meldet sich an Board! 10 % - hisst den Yellow Jack, Zweifüßer `Aye` - Trunkenbold Jack Sparrow meldet sich an Board! X - in der Kiste, Zweifüßer ``

`print()`	Ruft die `toString()` Methode auf und gibt den String auf die Konsole aus

Klasse Schiff

Attribute

Tabelle 2. Beschreibung der Klasse Schiff

Attribute	Beschreibung
`kanonen`	Die Anzahl der Kanonen
`typ`	Character, der den Typ des Schiffs festlegt (siehe Details Schiffe)
`kapitaen`	Der erste Pirat an Bord ist der Kapitän
`pirat2`	Zweiter Pirat
`pirat3`	Dritter Pirat
`pirat4`	Vierter Pirat
`pirat5`	Fünfter Pirat
`pirat6`	Sechster Pirat

Details Schiffe

Character	Bezeichnung	Maximalbesatzung	Kanonen
`s`	Schaluppe	3	1
`b`	Brigg	4	2
`f`	Fregatte	5	5
`g`	Galeone	6	8

Methoden

Sämtliche Get- und Set- Methoden sind wie üblich zu implementieren. Die restlichen Methoden sowie durchzuführende Plausibilitätsprüfungen sind in der nachstehenden Tabelle beschrieben.

Methoden	Beschreibung
`Konstruktor`	• Der Konstruktor ruft wie üblich die Methode `setTyp(typ)` auf. • Zusätzlich soll der Konstruktor noch die Methode `kanonenKlarmachen()` aufrufen.
`kanonenKlarmachen()`	Folgende Werte sollen abhängig vom Typ für die Kanone gesetzt werden: • Eine Schaluppe hat **eine** Kanone • Eine Brigg hat **zwei** Kanonen • Eine Fregatte hat **fünf** Kanonen • Eine Galeone hat **acht** Kanonen • Alle anderen Werte sind nicht erlaubt und es soll folgende Meldung ausgegeben werden: `Arrgh! Schaebige Schnigge vom Typ "TYP" ist nicht wuerdig gesegelt zu werden!`

Methoden	Beschreibung
`setTyp(typ)`	• Aus der Tabelle Details Schiffe können Sie die gültigen Wert entnehmen. • Werden ungültige Werte übergeben, so soll folgende Meldung auf die Konsole ausgegeben werden: `Harr! Der Schiffstyp TYP ist zu raeudig!`

`anheuern(pirat)`	• Übernimmt den Parameter `pirat` und setzt den Wert bei einem freien Piraten.
	• Der allererste Pirat ist immer der Kapitän, danach sollen alle anderen Piraten befüllt werden.
	• Folgende Anforderungen sind einzuhalten:
	  • Achten Sie auf maximale Anzahl an Piraten pro Typ Details Schiffe
	  • Eine Schaluppe darf nur **3** Piraten (inkl. Kapitän) aufnehmen
	  • Eine Brigg darf nur **4** Piraten (inkl. Kapitän) aufnehmen,
	  • usw…
	• Ist die maximale Anzahl an Piraten erreicht, dann soll folgende Meldung ausgegeben werden:
	`Arrg! Kein Grog mehr fuer den neuen Trunkenbold! Voll besetzt! Typ= TYP`

`print()`	Die Methode ruft die `toString()` Methode auf und gibt den `String` auf die Konsole aus.

Methoden	Beschreibung

`toString()`
`String`
	• Gibt einen `String` mit Informationen über das Schiff und der Besatzung zurück.
	Format:
	
	Beispiel:
	
	Listing 1. String einer Schaluppe
	`**********`
	`Crew Schaluppe (3): ❶`
	`----------`
	❷
	`Kapitän: `Aye` - Trunkenbold Oneeye Joe meldet sich an Board! 100%, Zweifüßer`
	`Pirat 2: `Aye` - Trunkenbold P2 meldet sich an Board! 100%, Zweifüßer`
	`Pirat 3: `Aye` - Trunkenbold P3 meldet sich an Board! 100%, Zweifüßer`
	`**********`
	❶ Statt dem Attribut Typ `typ` (z.B.: `s` ist gleich Schaluppe) soll die Bezeichnung ausgegeben werden. Weiters soll die maximale Größe in Klammer stehen. z.B.: (3) (siehe Details Schiff)
	❷ Verwenden Sie die `toString()` Methode des Piraten für die Ausgabe Detailinformationen zu den Piraten
	
	Listing 2. String einer Galeone
	`**********`
	`Crew Galeone (6):`
	`----------`
	`Kapitän: `Aye` - Trunkenbold Oneeye Joe meldet sich an Board! 100%, Zweifüßer`
	`Pirat 2: `Aye` - Trunkenbold P2 meldet sich an Board! 100%, Zweifüßer`
	`Pirat 3: `Aye` - Trunkenbold P3 meldet sich an Board! 100%, Zweifüßer`
	`Pirat 4: `Aye` - Trunkenbold P4 meldet sich an Board! 100%, Zweifüßer`
	`Pirat 5: `Aye` - Trunkenbold P5 meldet sich an Board! 100%, Zweifüßer`
	`Pirat 6: `Aye` - Trunkenbold P6 meldet sich an Board! 100%, Zweifüßer`
	`**********`

Getränkeüberblick

Character	Beschreibung
`w`	Wasser - Vermindert die Gesundheit um 10 Prozent
`g`	Grog (Mischung aus Wasser und Rum) - Erhöht die Gesundheit um 10 Prozent
`r`	Rum - Erhöht die Gesundheit um 20 Prozent

Merke auf

- Neue Objekte (Instanzen) können in Java mit dem `new` Operator erzeugt werden
- In den Klammern werden die Parameterwerte für den Konstruktor übergeben.

```
public void erzeugePirat() {
    Pirat pirat1 = new Pirat("Blackbeard", 100, true);
    ...
}
```

- Aufruf der Methode `toString()` eines Piraten aus der `toString()` Methode einer Schiff Instanz

```
public class Schiff {

    private Pirat pirat1;

    public String toString() {

        String info;
        ...

        // Aufruf der Methode toString() am Objekt pirat1 und
Zuweisung des Rückgabewertes zum String info
        info = pirat1.toString();

        ...
    }

}
```

- Prüfungen mit dem Datentyp `char`

```
public class Pirat {

    public void trinken(char getraenk) {
        if(getraenk == 'w') {
            ...
        } else {
            ...
        }
    }

}
```

- Beispiel einer Testklasse

```
public class TestPirat {

    public void testKielholen() {

        // Erzeugen einer Piraten Instanz
```

```
        Pirat p1 = new Pirat("Oneeye Joe", 100, false);

        // Ausgabe des Piraten auf die Konsole
        System.out.println( p1.toString() );

        // Aufruf kielholen() - Gesundheit sollte verringert
werden
        p1.kielholen();

        // Überprüfen der Gesundheit durch Ausgabe auf die
Konsole
        System.out.println( p1.toString() );

        // usw...
        p1.kielholen();
        System.out.println( p1.toString() );
        ...

    }

    public void testTrinken() {
        // Erzeugen eines Piraten Objektes
        Pirat p1 = new Pirat("Oneeye Joe", 50, false);

        p1.trinken('w');
        System.out.println( p1.toString() );

        // Alle Werte testen, auch ungültige Werte!
        p1.trinken('x');
        System.out.println( p1.toString() );
        ...
    }

}
```

Themen

- `if/else` Prüfungen
- Erzeugen neuer Instanzen einer Klasse
- Methodenaufrufe eigener Objekte
- Methodenaufrufe fremder Objekte
- Testklassen
- `toString()` Methode