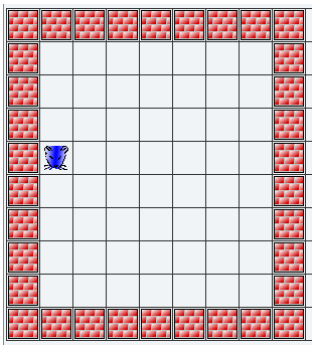
# Klassenarbeit über den Java-Hamster

**Aufgabe 1** Was versteht man unter einer Prozedur oder einer Funktion?

**Aufgabe 2** Welche Vorteile bietet der Einsatz von Prozeduren.

**Aufgabe 3** Schreibe eine Prozedur, die den Hamster eine 360° Drehung vollführen lässt.

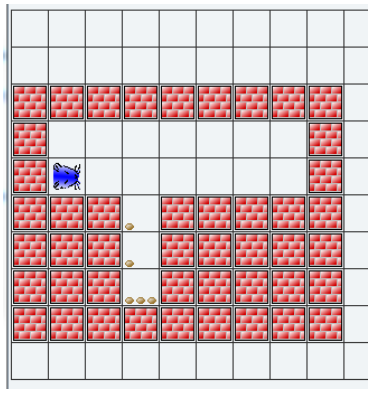
**Aufgabe 4** Schreibe ein Programm, das den Hamster in jede Ecke des Spielfelds ein Korn legen und an die Ausgangsposition zurückkehren lässt. Zu Beginn hat der Hamster 4 Körner im Maul. Nutze lediglich die **For-Schleife**.



**Aufgabe 5** Worin unterscheiden sich allgemein die For-Schleife und die While-Schleife?

**Aufgabe 6** Ändere dein Programm aus Aufgabe 4 so, dass alle For-Schleifen durch While-Schleifen ausgetauscht werden.

**Aufgabe 7** Finde die Fehler in der Programmierung.
Die Aufgabe lautet wie folgt: Der Hamster steht vor einem mit Körner gefülltem Loch. Er weiß nicht, wie tief das Loch ist. Er soll alle Körner aufsammeln und zum Ausgangspunkt zurückkehren.



```
void main()

        rechtsFreivor();
        kornDageh();
        umkehr();
        rechtsFreivor();
        umkehr();
        while (vornFrei())
        {
        vor()
        }
        umkehr();
}

//rechtsFrei
boolean rechtsFrei()
{
rechtsUm();
if (vornFrei(
{
linksUm();
return true;
}
else
{
linksUm();
return false;
}
}

//rechtsUm
void rechtsUm()
{
        for (int i=1; i<=4; i++)
        {
        linksUm();
        }
}
```

```
//umkehr
void umkehr()
{
linksUm();
linksUm();
linksUm();
}

//rechtsFrei und vor
void rechtsFreivor()
{
        while !(rechtsFrei())
        {
        vor();
        }
        rechtsUm();
}

//Korn da, dann nimm und geh
void kornDageh()
{
        while (!vornFrei())
        {
        vor();
        while (kornDa())
        {nimm();

        }
}
```

**Klassenarbeit über den Java-Hamster**

**Aufgabe 1** Was versteht man unter einer Prozedur oder einer Funktion?

Eine Prozedur oder eine Funktion kann als Unterprogramm verstanden werden, mit dem ein neuer Befehl programmiert werden kann. Prozeduren sind dabei Handlungsanweisungen, während Funktionen einen Ausgabewert besitzen.

**Aufgabe 2** Welche Vorteile bietet der Einsatz von Prozeduren.

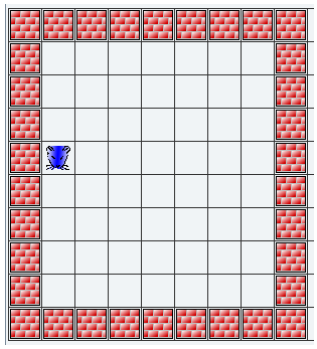
Der Vorteil des Einsatzes von Prozeduren liegt in einer besseren Übersichtlichkeit von Programmen, in separaten Lösungen von Teilproblemen und Platzeinsparungen, sowie in einer einfacheren Fehlerbehebung, und Wiederverwendbarkeit.

**Aufgabe 3** Schreibe eine Prozedur, die den Hamster eine 360° Drehung vollführen lässt.

```
Void Drehung()
{
linksUm();
linksUm();
linksUm();
linksUm();
}
```

(oder durch FOR-Schleife)

**Aufgabe 4** Schreibe ein Programm, das den Hamster in jede Ecke des Spielfelds ein Korn legen und an die Ausgangsposition zurückkehren lässt. Zu Beginn hat der Hamster 4 Körner im Maul. Nutze lediglich die **For-Schleife**.



```
void main()
{

        for (int i=1; i<=4; i++)
        {
        vor();
        }
        linksUm();
        gib();
        for (int i=1; i<=6; i++)
        {
        vor();
        }
        linksUm();
        gib();
        for (int i=1; i<=7; i++)
        {
        vor();
        }
        linksUm();
        gib();
        for (int i=1; i<=6; i++)
        {
        vor();
        }
        linksUm();
        gib();
        vor();
        vor();
        vor();
}
```

**Aufgabe 5** Worin unterscheiden sich allgemein die For-Schleife und die While-Schleife?

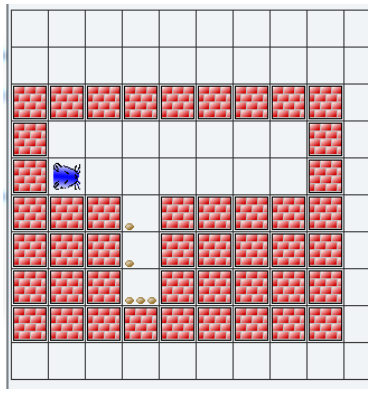
Die FOR-Schleife wird durch eine Variable initialisiert, während die While-Schleife durch eine Bedingung initialisiert wird.

**Aufgabe 6** Ändere dein Programm aus Aufgabe 4 so, dass alle For-Schleifen durch While-Schleifen ausgetauscht werden.

```
void main() {

		while (!maulLeer())
		{
		Wandundkorn();
		}
		vor();
		vor();
		vor();

}


//bis zur Wand und noch ein Korn ablegen
void Wandundkorn()
{
		while (vornFrei())
		{
		vor();
		}
		gib();
		linksUm();
}
```

**Aufgabe 7** Finde die Fehler in der Programmierung.

Die Aufgabe lautet wie folgt: Der Hamster steht vor einem mit Körner gefülltem Loch. Er weiß nicht, wie tief das Loch ist. Er soll alle Körner aufsammeln und zum Ausgangspunkt zurückkehren.



```
void main()
{
		rechtsFreivor();
		kornDageh();
		umkehr();
		rechtsFreivor();
		umkehr();
		while (vornFrei())
		{
		vor();
		}
		umkehr();
}

//rechtsFrei
boolean rechtsFrei()
{
rechtsUm();
if (vornFrei())
{
linksUm();
return true;
}
else
{
linksUm();
return false;
}
}

//rechtsUm
void rechtsUm()
{
		for (int i=1; i<=3; i++)
		{
		linksUm();
		}
}
```

```
//umkehr
void umkehr()
{
linksUm();
linksUm();
}

//rechtsFrei und vor
void rechtsFreivor()
{
		while (!rechtsFrei())
		{
		vor();
		}
		rechtsUm();
}

//Korn da, dann nimm und geh
void kornDageh()
{
		while (vornFrei())
		{
		vor();
		while (kornDa())
		{nimm();
		}
		}
}
```