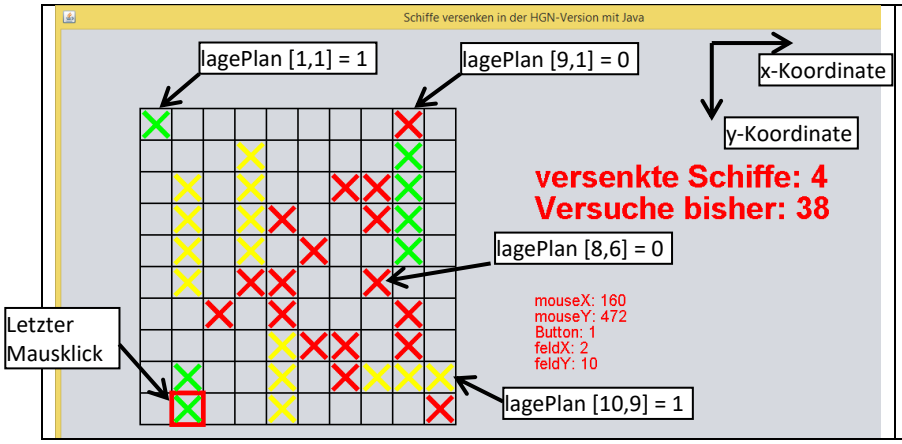
# Projekt 8: Schiffe versenken – Speichern im zweidimensionalen Array

Beim Spiel „Schiffe versenken“ werden auf einer Spielfläche von 10 mal 10 Feldern insgesamt 10 Schiffe versteckt. Ein Schlachtschiff (5 Kästchen), zwei Kreuzer (je 4 Kästchen), drei Zerstörer (je 3 Kästchen) und vier U-Boote (je 2 Kästchen). Aufgabe des Spielers ist es, die Schiffe zu finden, wobei mit der Maus auf je ein Feld geklickt wird und Java zurückmeldet, ob ein Schiff getroffen wurde: In der Grafik bedeutet grün ein Treffer, rot ein Versuch ins Leere und gelb ein komplett versektes Schiff. Um die Lage der Schiffe speichern zu können, benötigt man ein **zweidimensionales Array**: Für jede x- und für jede y-Koordinate zwischen 1 und 10 wird in einem Array Namens **lagePlan** gespeichert, ob dort ein Teil eines Schiffes liegt (Wert 1) oder nicht (Wert 0). Der Befehl lagePlan [9,1] = 0 bedeutet: an der Stelle x = 9 (9. Spalte) und y = 1 (1. Zeile) liegt kein Teil eines Schiffes, das gesetzte Kreuz ist also rot (s. Grafik Unten). Um das zweidimensionale Array anzulegen, benutzt man die Befehle wie rechts neben der Grafik angegeben: Die doppelte Klammer [][] steht für ein zweidimensionales Array, die Angabe 11 wird benötigt, damit Plätze von 0 bis 10 reserviert werden, der Platz 0 wird im Spiel jedoch nicht benötigt. Der Accesss-Modifier **public static** bedeutet, dass das Variablenfeld in allen Klassen des Projekt benutzt werden darf (**public**) und es dafür auch nur eine einzige Version gibt (**static**): so wird erreicht, dass auch in verschiedenen Objehkten der Klasse auf die gleichen variablenwerte zugegriffen wird, z. B. **Gui.lageplan [5][6] = 1**;.




**Anlegen des zweidimensionalen Arrays lageplan:**

public static int [][] lageplan;
lageplan = new int [11][11];

**Bedeutung der Farbkreuze:**

Rot: kein Treffer
Grün: Treffer
Gelb: Schiff versenkt

**Aufgabe 1:** Lade das Projekt **SchiffeVersenken** aus dem Lehrerverzeichnis herunter. Die Klasse **SchiffeVersenken** beinhaltet lediglich die Spielregeln und einen Button zum Starten des Spieles, hier musst du nichts verändern. Die Klasse **Gui** beinhaltet bereits verschiedene Methoden, die zum Zeichnen der Spielfläche und zur Reaktion auf die Mausbewegung notwendig sind. In einem Objekt der Klasse **Schiff** wird jeweils die Lage eines der 10 Schiffe angelegt und gespeichert.
Ergänze die Methode **zeichnen()** in der Klasse **Gui** so, dass bei einem Mausklick ein rotes Rechteck in dem Kästchen gezeichnet wird, auf das der Mauszeiger zeigt. Wie du in der oberen Grafik und beim Ausführen des Spieles erkennen kannst, werden bei jedem Mausklick bereits vier Variablen berechnet: **mouseX** und **mouseY** sind die x- und y-Koordinaten des Mauszeigers beim Klick. Ganz am Ende der Klasse Gui erkennst du, dass dazu in dem **Mouseadapter** auf einen Mausklick regiert wird: mit dem Befehl **repaint()** wird die Methode **paint()** aufgerufen, die du weiter oben findest. In dieser Methode **paint()** wird über den Befehl **location.getX()** die x-Koordinate und analog die y-Koordinate des Mauszeigers abgefragt. Direkt anschließend wird auf die Koordinaten der Kästchen des Spielfeldes umgerechnet. **feldX** und **feldY** beinhalten diese Koordinaten und können Werte zwischen 1 und 10 annehmen (s. Oben). Um ein Rechteck innerhalb der Methode **paint()** zu zeichnen, benutzt du die schon fertige Methode **public void rechteck (int x1, int y1)**. **x1** und **y1** sind die Kästchenkoordinaten eines Kästchens zwischen 1 und 10. Der Befehl **rechteck (1,1);** setzt z. B. ein Rechteck im linken, oberen Kästchen.

**Aufgabe 2:** Beim Start des Programms werden die Lage der 10 Schiffe per Zufall gesetzt. In dem Variablenfeld **lagePlan [x,y]** haben alle Kästchen, in denen sich ein Teil eines Schiffes befindet, den Wert 1, wie es bereits oben erklärt wurde. Beim Start des Programms soll in allen Kästchen, in denen ein Teil eines Schiffes liegt, ein grünes Kreuz gezeichnet werden. Benutze dazu die fertige Methode **public void Kreuz (int x1, int y1)**, die ganz analog zur Methode **rechteck** aus Aufgabe 1 funktioniert. **Tipp:** Benutze zwei Schleifen, um alle x- und y-Koordinaten nacheinander zu behandeln. Mit dem Befehl **f2.setColor(Color.RED);** kannst du die Zeichenfarbe für folgende Zeichenoperationen auf rot setzen. Andere Farbkonstanten sind z. B. **YELLOW** und **GREEN**. Mit

dem Befehl **f2.setStroke(linieDick);** kannst du Dicke der Linien verbreitern, eine andere Konstante für die Linienstärke ist **linieDünn**.

**Aufgabe 3:** Aufgabe 2 wird natürlich nicht beim fertigen Spiel vorkommen, da die Lösung dadurch direkt vorgegeben würde, setze mit dem Kürzel „**//**“ die entsprechenden Befehle wieder auf inaktiv. In dieser Aufgabe soll auf einen Mausklick hin im entsprechenden Kästchen jeweils ein grünes Kreuz (Treffer) oder rotes Kreuz (kein Treffer) gesetzt werden. Dazu wird ein weiteres zweidimensionales Array **spielFeld** benötigt, in dem der jeweilige Spielstand gespeichert wird. Folgende Werte sind möglich:

| Wert für ein Kästchen im Array **spielFeld** | Bedeutung |
|---|---|
| 0 | Das Kästchen beinhaltet **keinen** Teil eines Schiffes, auf das Kästchen wurde **noch nicht geklickt**, es wird also kein Kreuz gezeichnet |
| 1 | Das Kästchen beinhaltet **einen** Teil eines Schiffes, auf das Kästchen wurde **noch nicht geklickt**, es wird also kein Kreuz gezeichnet |
| 2 | Das Kästchen beinhaltet **keinen** Teil eines Schiffes, auf das Kästchen **wurde bereits geklickt**, es wird also ein rotes Kreuz gezeichnet |
| 3 | Das Kästchen beinhaltet **einen** Teil eines Schiffes, auf das Kästchen **wurde bereits geklickt**, es wird also ein grünes Kreuz gezeichnet |

Beispiel: **spielFeld [4,5] = 3;** bedeutet: Das Kästchen mit der x-Koordinate 4 und der y-Koordinate 5 beinhaltet einen Teil eines Schiffes und wurde bereits mit Mausklick untersucht, es muss also ein grünes Kreuz gezeichnet werden.

Überlege, in welchen Fällen die Werte des Arrays **spielFeld** nach einem Mausklick veändert werden müssen. Programmiere die einzenen Veränderungen innerhalb der Methode paint(), bevor die Kreuze gezeichnet werden. **Tipp:** Spätestens jetzt werden nach jedem Mausklick alle Kreuze neu gezeichnet, dies geht so schnell, dass der Benutzer nichts davon merkt.

**Aufgabe 4:** Die Methode **versenkt()** in der Klasse **Schiff** untersucht, ob ein Objekt der Klasse Schiff bereits vollständig gefunden wurde, beim Spiel Schiffe versenken sagt man in diesem Moment „Schiff versenkt“. Wurde das Schiff tatsächlich bereits versenkt, setzt die Methode in allen Kästchen, die Teil des Schiffes sind, im Array **spielFeld** den Wert 4. Überprüfe innerhalb der methode paint() durch Aufruf der Methode versenkt() für **jedes** Schiff, welche Schiffe bereits versenkt wurden und setze gelbe Kreuze bei allen Schiffen, die versenkt wurden.

**Aufgabe 5:** Während des Spieles sollen die Versuche, also die Mausklicks auf ein Kästchen, gezählt werden. Außerdem soll jeweils angezeigt werden, wie viele Schiffe bisher versenkt wurden. Sind alle 10 Schiffe versenkt, soll eine entsprechende Meldung ausgegeben werden.

**Zusatzaufgabe 1:** Der Konstrukor **public Schiff(int pLaenge)** der Klasse **Schiff** setzt ein Schiff mit der Länge **pLaenge** auf der Zeichenfläche, d. h. es setzt im Array **lagePlan** den Wert 1 bei allen Kästchen, die Teil des Schiffes sind. Dabei müssen folgende Regeln eingehalten werden:
Ein Schiff liegt entweder parallel zur X-Achse oder parallel zur Y-Achse, zwei Schiffe haben keine gemeinsame Kästchen, 2 Schiffe dürfen niemals aneinanderstoßen. Betrachte folgendes Beispiel für ein Schiff mit der Länge 4: Zu Beginn werden folgende Variablenwerte zufällig gezogen: posX = 3; posY = 4; richtung = 1;
Beschreibe im Heft, auf welchen Kästchen untersucht werden muss, ob dort bereits ein anderes Schiff liegt? Erkläre im Heft, welche Bereiche dabei von der Methode **pruefePos** nacheinander untersucht werden.

**Zusatzaufgabe 2:** Erkläre im Heft, wie die Methode **versenkt()** funktioniert.

**Zusatzaufgabe 3:** Erkläre mit eigenen Worten im Heft, welchen Vorteil es bringt, neben der Klasse **Gui** die Klasse **Schiff** einzuführen. Gäbe es eine andere Möglichkeit der Speicherung, die mit einer Klasse auskommt? Begründe, welche der beiden Alternativen du bevorzugen würdest, wenn du das Spiel komplett programmieren würdest.