1) Informiere dich mithilfe des Informationsblattes „Arrays“ oder Internets was Arrays sind. Erkläre schriftlich in der dir auf IServ zugewiesenen Gruppe in eigenen Worten die Funktionsweise von Arrays an einem selbstgewählten Beispiel.

Hinweis: Deine Erklärung soll unter anderen folgendes enthalten: Deklaration von Arrays als Variable und Objekt, wie kann Array durchlaufen werden, funktionsweise der Schleife (Code angeben), …

Tipp: Es lohnt sich bereits Inhalte für die Implementierung vorzubereiten ☺

Die Bearbeitung von 2 und 3 erfolgt erst nach der Pause und der Präsentation

2) Lade dir erneut die Vorlage aus IServ herunter.
a) Erstelle die Array-Variable hindernisse. Befülle das Array mit 25 Objekten mithilfe der for-Schleife im Konstruktor.

b) Implementiere die Methode gibHindernisIndex() vom Datentyp int. Mithilfe der Zähleschreife soll überprüft werden, ob der Ball mit den Hindernissen kollidiert. Ist das der Fall so soll Index zurückgegeben werden, wenn nicht so soll -1 zurückgegeben werden.

c) Die Methode void fuehreAus() muss angepasst werden. Damit wir nicht jedes einzelne Objekte überprüfen müssen, ob es noch vorhanden ist, oder nicht, muss hierfür die Methode index = gibHindernisIndex() implementiert werden. Mithilfe des Index wird überprüft, ob ein Hindernis an der Stelle des Index noch vorhanden ist oder nicht. Ist ein Hindernis getroffen worden, so soll die Richtung geändert werden und das Hindernis an der Position des Index mit der Methode set.Hidden(true) versteckt werden.

3) Bereite dich darauf vor, deine Ergebnisse zu präsentieren.

Tipps findest du auf den nächsten Seiten.

Tipps zu der Bearbeitung der Aufgabe 2 a)

Implementierung der for-Schleife

Tipp 1:
Befülle die for-Schleife wie folgt:

```
hindernisse[i] = new Rectangle(40 + i*20, 100, 10, 10, Color.orange)
```

Tipp 2: Vervollständige die for-Schleife

```
hindernisse = new Rectangle[25];
   for (int i = 0; i < ; i++){
      hindernisse[ ] = new (40 + i*20, 100, 10, 10, Color.orange);
   }
```

Tipps zu Aufgabe 2b finden sich auf der nächsten Seite.

Tipps zu der Bearbeitung der Aufgabe 2 b)

Implementierung der Methode gibHindernisIndex()

Tipp 1:

Auch diese Methode implementieren wir durch direkte Umsetzung des umgangssprachlichen Algorithmus' in Java. Die Anzahl der Elemente eines arrays lässt sich mit length abfragen. Man könnte natürlich auch direkt 25 schreiben, weil wir 25 Hindernisse haben. Die aktuelle Länge abzufragen hat den Vorteil, dass das Programm an dieser Stelle nicht geändert werden muss, wenn wir die Anzahl der Hindernisse verändern.

```
index = -1
wiederhole für i in (0..24)
  falls ball berührt hindernisse[i]
    index = i
Rückgabe index
```

Tipp 2: Unvollständiger Code

```
int index = -1;
    for (int i = 0; i<; ){
      if ( (hindernisse[i])){
        index = i;
```

Tipp 3: Unvollständiger Code

```
int index = -1;
    for (int ; i< ; ){
      if (ball.intersects(hindernisse[i])){
        index = i;
    }
}
```

Tipps zu Aufgabe 2c finden sich auf der nächsten Seite.

Tipps zu der Bearbeitung der Aufgabe 2 c)

3. Implementation von int index = gibHindernisIndex();

Da wir bei jedem Durchlauf der Animationsschleife für alle Hindernisse überprüfen müssen, ob eine Kollision mit dem Ball stattgefunden hat, deklarieren wir dafür die Methode int gibHindernisIndex(), die entweder den Index des Hindernisses zurückgibt, das vom Ball berührt wird oder -1, falls keine Kollision erfolgt ist.

Tipp 1:

falls index ist ungleich -1
aendereRichtung2
verstecke hindernisse an der Stelle des Index;

Tipp 2: Unvollständiger Code

```
int = gibHindernisIndex;
  if (index!=-1){

     [index].setHidden();
 }
```

Tipp 3: Ausschnitt des Codes aus der while-Schleife der void-Methode fuehreAus()

```
  if (ball.intersects(schlaeger)){
    aendereRichtung(2);
  }
  int index = gibHindernisIndex();
  if (index!=-1){

     [index].setHidden(true);
 }
  ball.move(5);
  view.wait(20);
}
```