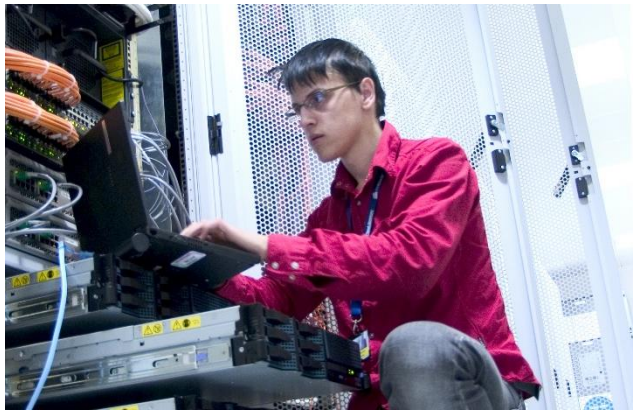
EMIL-VON-BEHRING GYMNASIUM

Erzeugung von Nebenläufigkeit



Erzeugung von Nebenläufigkeit

- Thread ist eine von Java vorgegebene Klasse, von welcher man erbt (kein Import nötig!). https://docs.oracle.com/javase/8/docs/api/java/lang/Thread.html
- Die Methode public void run() muss aufgrund des Interfaces Runnable implementiert werden, welches die Klasse Thread implementiert.
- Der Code in der run-Methode des Thread-Objekts kann parallel ausgeführt werden, indem man Thread.start() aufruft.
  - (Bei Thread.run() wird kein neuer Aktivitätsfaden erzeugt, sondern der „main-Thread“ führt diese Methode (dann sequentiell) aus.)
- Mit Thread.join() kann der „main-Thread“ die gestarteten, parallel laufenden Threads wieder „einsammeln“ bzw. die Aktivitätsfäden wieder zusammenführen.

Beispielvorlage

```
public class MyProgramm {

    public void myMethod() {
        // Erzeugen des Threads
        Thread t = new MyThread();

        t.start();
// Thread wird in Verwaltung aufgenommen
// Sobald er an der Reihe ist wird mit
Ausführung der Methode run begonnen
        try{
        t.join();
        }catch(InterruptedException e){}
// main-Thread wartet bis die run-Methode
fertig durchgelaufen ist
// Falls der Thread die run-Methode nicht
beenden kann, wird eine Exception geworfen
    }
}
```

```
class MyThread extends Thread {
    // Nötige Attribute,
    Konstruktor(en),
    // Hilfsmethoden usw.

    /**
     * Hier beginnt der Thread seine
    Arbeit
     */
    public void run() {
        // Arbeitsauftrag des Threads
    }
}
```

Arbeitsauftrag

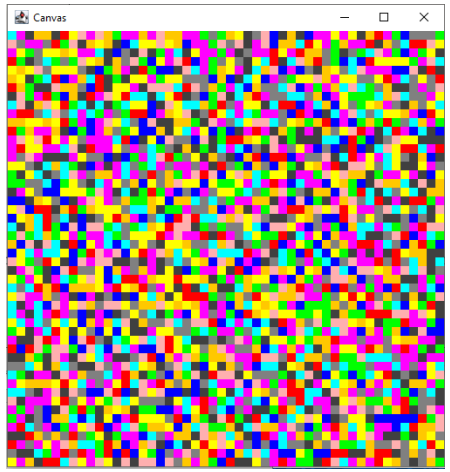
- Erstelle ein neues BlueJ-Projekt Threads mit den Klassen **MINUS**, **PLUS** und **TEST** wie folgt:
- **MINUS** und **PLUS** erben von der Klasse Thread
- In der run()-Methode wird jeweils 100 Mal (for-Schleife) die folgende Anweisung ausführt:
  - Klasse Minus: System.out.println(„-“)
  - Klasse Plus: System.out.println(„+“)
- In der Klasse **TEST** wird dann jeweils ein Objekt der Klassen erzeugt und der zugehörige Thread gestartet und
- **Führe den Code mehrfach hintereinander aus**. Was kannst du bezüglich der Ausgabe dieses parallelen Programms feststellen?
  → sehr wichtige Erkenntnis

Aufgabe – Robo Dance

- Öffne das Projekt Robo Dance. Implementiere in der Klasse TANZPARTY:
  - TANZPARTY erstellt für beliebig viele TANZROBOTER jeweils einen eigenen ROBOTERTHREAD, welcher seinen TANZROBOTER parallel zu den anderen steuert. Dabei soll jeder TANZROBOTER entsprechend oft die Methode tanzen() ausführen. → siehe Parameter des Konstruktors
  - Schreibe die Klasse ROBOTERTHREAD und überlege dir, welche Daten jedem Thread über den Konstruktor von ROBOTERTHREAD mitgeben werden müssen.
  - Erstelle in der TANZPARTY alle Objekte und lass die Roboter parallel tanzen.

  - Tipp: Man kann die Bewegung verlangsamen, indem man eine Wartezeit mit dem Befehl (Thread.sleep(100) mit try-catch) zwischen Schritten bei jedem Thread einbaut.

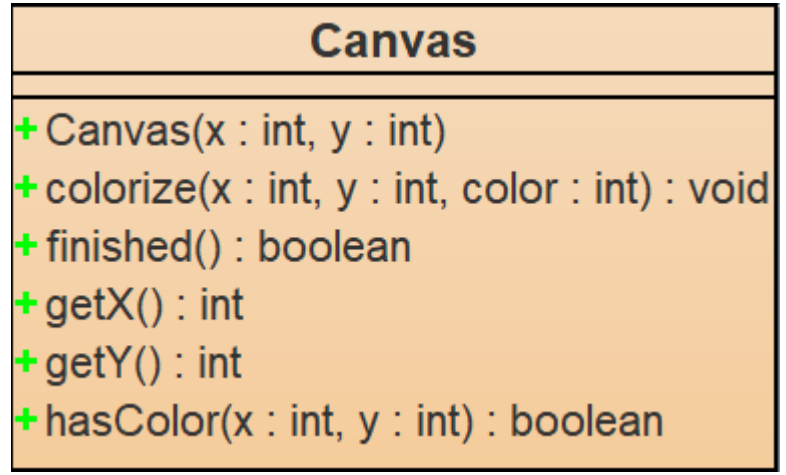
Aufgabe – Picasso



- Es soll mit mehreren Maler-Threads eine Leinwand (canvas) parallelisiert bemalt werden.
- Es gibt eine gemeinsame Leinwand, die in der passenden Größe initialisiert werden soll.
- Jeder Maler-Thread hat eine Nummer/Zahl, welche seine Farbe darstellt. (0=rot, 1=grün, 2=blau, usw.)
- Die Maler-Threads sollen dabei so lange zufällig ausgewählte Felder/Pixel auf der Leinwand bemalen, bis die Leinwand fertig bemalt ist.
- → Hilfestellung auf der nächsten Folie

Picasso – Hilfestellung

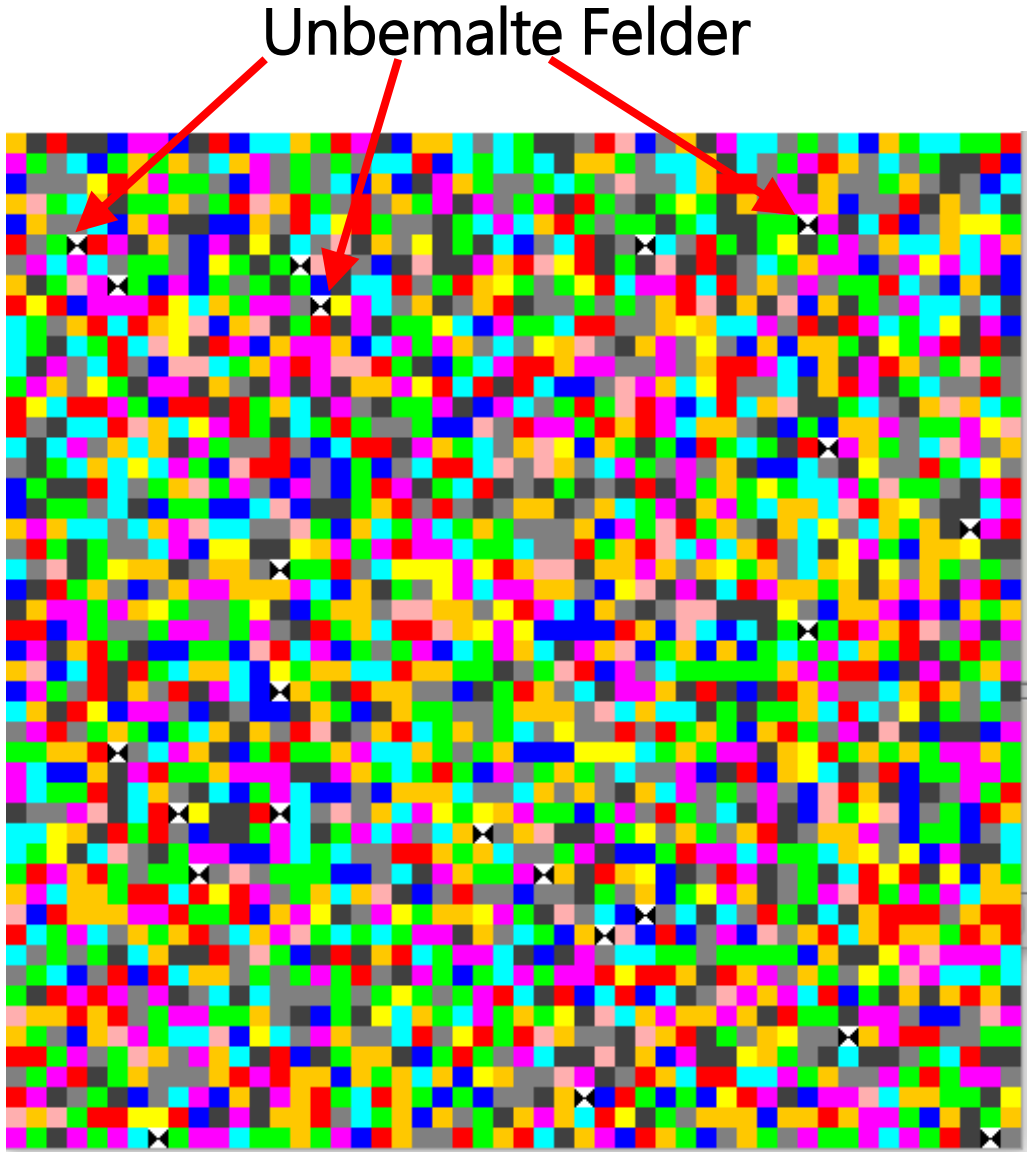
- Benutze die Methoden der Klasse Canvas:
  - Canvas(…): Kontruktor mit Angabe der Größe der Leinwand
  - colorize(…): Angabe des Pixels und der Farbnummer
  - finished(): ist Leinwand fertig bemalt?
  - hasColor(…): hat dort schon jemand anders gemalt?
  - getX(), getY(): Größe der Leinwand in x,y-Richtung



- Die Zufälligkeit muss mithilfe der Klasse Random von Java (java.util.Random) realisiert werden. Einfach ein neues Objekt der Klasse Random erzeugen (Standardkonstruktor)
  -> Methode **`nextInt(int bound)`** für zufällige ganze Zahlen von inklusive 0 bis exklusive **`bound`** → kein Math.random(…)!

Warum werden nicht alle Felder bemalt?

- Beim parallelen Bemalen der Leinwand kommt es zu einer Wettlaufsituation, wenn zwei Threads gleichzeitig dasselbe zufällige Feld auswählen – welches noch nicht bemalt ist – und beide dieses Feld mit ihrer Farbe bemalen.
- Dadurch bleibt ein **anderes** Feld unbemalt, weil sich die Leinwand nach Breite*Höhe-vielen Bemalungen als fertig bemalt ansieht.
- Welche Methoden verursachen den Fehler?



Wettlaufsituation Robo Dance

- In der Methode tanzen() des Tanzroboters werden zufällig eine dieser drei Methoden aufgerufen.
- Können Tanzroboter aufgrund einer Wettlaufsituation ineinander krachen?
- Ja, aufgrund von der Verzahnung von IstRoboter() mit Schritt() bei verschiedenen Tanzroboterthreads

```
public void umdrehen() {
    LinksDrehen();
    LinksDrehen();
}
public void schritt() {
    if(IstWand()) {
        umdrehen();
    } else {
        if(IstRoboter()) {
            LinksDrehen();
        } else {
            Schritt();
        }
    }
}
public void schrittNachRechts() {
    RechtsDrehen();
    if(!IstWand() && !IstRoboter()) {
        Schritt();
    }
}
```