Allgemein:

- ExecutionMonitor(Debugger) startet eine JVM mit TestsuitExecuter, dieser führt die Tests aus

Klassen:

- Klasse "TestsuitExecuter":
  - Das ist die Anwendung, die von unserem Debugger beobachtet wird
  - bekommt eine Liste von ".class" Files (die Testklassen) und führt alle Testfälle dieser Klassen aus
    - Anmerkung: man kann selbst bestimmen, was zwischen den Ausführungen von einzelnen Tests passiert d.h. Es werden nicht alle Tests aufeinmal abgearbeitet (durch Verwendung von 'Markierungsmethoden', bekommt unserer Debugger Metainformation z.B. ob der gerade abgelaufende Testfall erfolgreich war oder nicht)
- Klasse "JVMConnectionCreator":
  - Startet eine JVM bzw. Den TestsuitExecuter
- Klasse "Edge":
  - Repräsentiert einen Event
  - ist wie besprochen
  - enthält zusätzlich den Eintrag "enteredFromMethod":
    - enthält die Methode, die die aktuelle Methode aufgerufen hat
    - diese Information ist nur bei Edges vorhanden, die lineFrom "-1" haben
    - falls die aufrufende Methode ein Junit Testfall ist --> Wert ist immer der String "TestCase"
    - für spätere Zwecke könnte das nötig sein z.B. Wenn man die einzelnen Graphen miteinander verbinden möchte
- Klasse "ExecutionMonitor":
  - ist unserer Debugger
  - protokoliert alle Methoden der zu testenden Anwendung
    - Konstruktor Aufrufe werden ignoriert --> kann man aber leicht mitprotokolieren
  - verwendet folgende Requests:
    - MethodeEntry: nur dafür da damit man feststellen kann, ob gerade eine neue Methode aufgerufen wurde (setzt nur einen Flag, die eingentliche Verarbeitung findet im StepRequest statt)
    - MethodExitRequest: wird verwendet, um den "abstrakten Call Stack"(Beschreibung weiter unten) abzubauen
    - StepRequest: das wichtigste hier z.B. Events für die Callgraph-Gruppe erzeugen
    - ExceptionRequest: der abstrakte Callstack wird mit dem eigentlichen Callstack synchronisiert
  - abstrakter Callstack:
    - das ist die Klasse " MethodExecutionLogger"
    - Repräsentiert den Callstack der zu testenden Anwendung aber enhält nur Methoden aus dem zu testenden Projekt
    - enthält auch einen Stack von "lineNumber"
      - pro Methode aus dem abstrakten Callstack gibt es einen Stackframe, der die letzte ausgeführte Zeilennummer in der entsprechenden Methode enthält
      - --> dadurch kann man lineFrom, lineTo angeben
- Klasse "Frontend":
  - für Testzwecke
  - man muss den Classpath und die Testmethoden-Namen hier angeben

- Classpath: Junit+Hamcrest , für diese Projekt, für das Testprojekt und alle dazugehörigen Libs