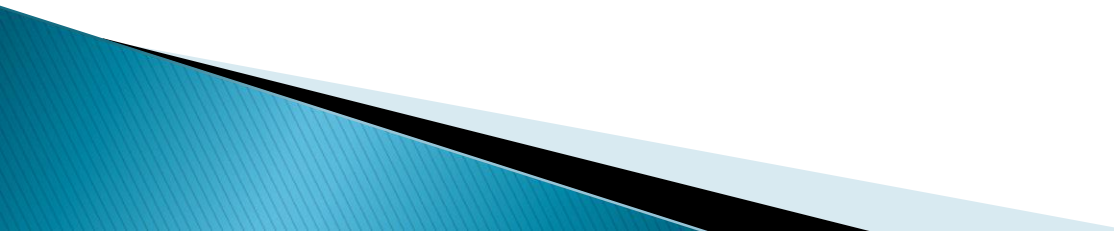


Gegenüberstellung der
wichtigen Begriffe

Inhaltsverzeichnis

- ▶ Spezifikation \leftrightarrow Verifikation
 - ▶ Algorithmus \leftrightarrow Programm
 - ▶ Prozess \leftrightarrow Multithreading \leftrightarrow Interprozesskommunikation
 - ▶ Value Type \leftrightarrow Reference Type
 - ▶ Stack-Speicher \leftrightarrow Heap-Speicher
 - ▶ Call by value \leftrightarrow Call by reference
- 

Spezifikation <-> Verifikation

Ausgangspunkt und Fundament für die Entwicklung eines Programmes oder einer Aufgabenstellung, welches mithilfe des Computers gelöst werden soll.

Spezifikation beginnt mit der Aufgabenstellung und Erarbeitung einer Spezifikation.

- Vollständigkeit
- Detailliertheit
- Eindeutigkeit

Sie ist eine der wichtigsten Aufgaben bei der Entwicklung von komplexer, umfangreicher Software und Sicherheitskritischer Anwendungen.

- Dynamisches Testen
- Statisches Testen

Algorithmus <-> Programm

Vorgehensweise (Lösungsplan) um ein bestimmtes Problem zu lösen. Es ist sehr wichtig in der Informatik, weil sie eine Grundlage für die konkrete Programmierung.

Beispiele:

Addiere
Subtrahiere
Multipliziere

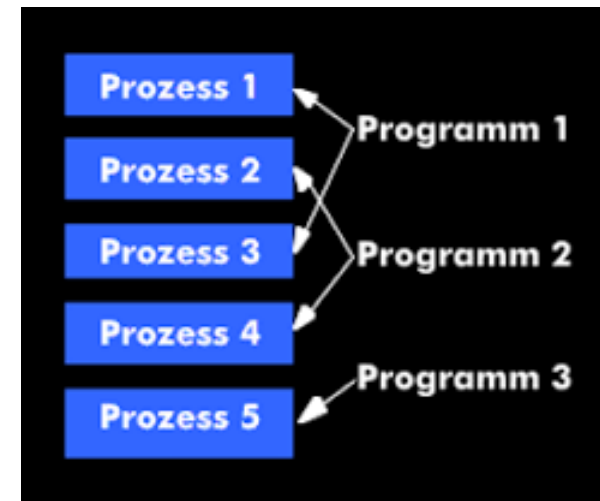
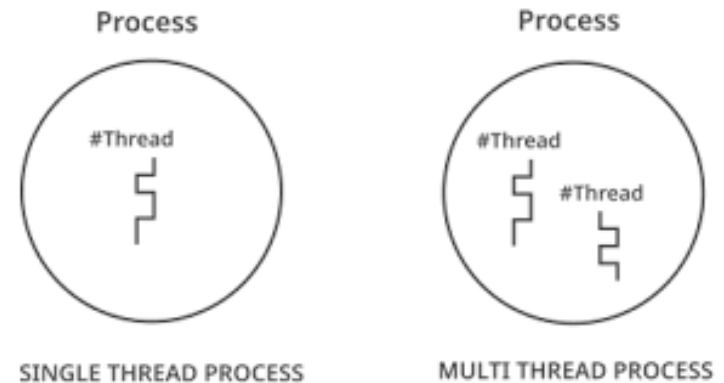
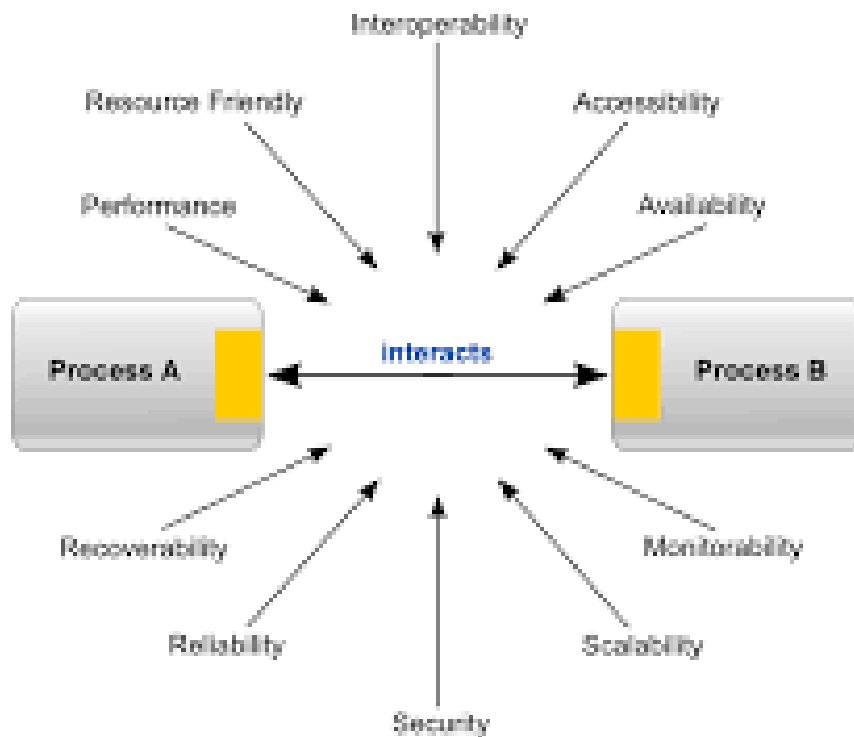
Ein Programm ist eine Folge von Quellcode, welches innerhalb eines Dateienbereichs oder mehrerer Dateienbereiche weitere kleinere Programme mit den dazugehörigen Algorithmen ausführt.

Beispiele:

Farbmischer
Taschenrechner
Kostenplaner

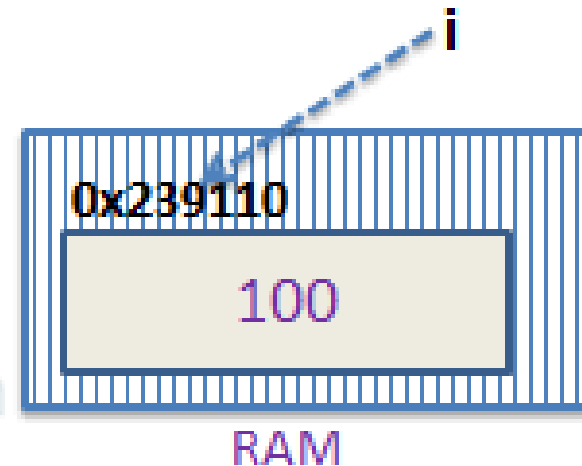
Prozess <-> Multithreading <-> Interprozesskommunikation

Interprocess Communication

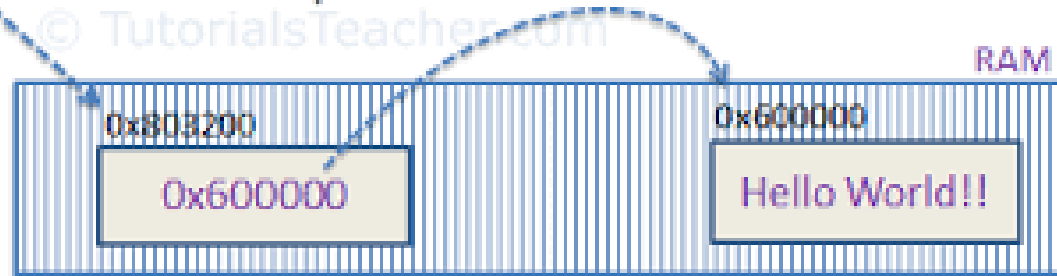


Value Type <-> Reference Type

```
int i = 100;
```



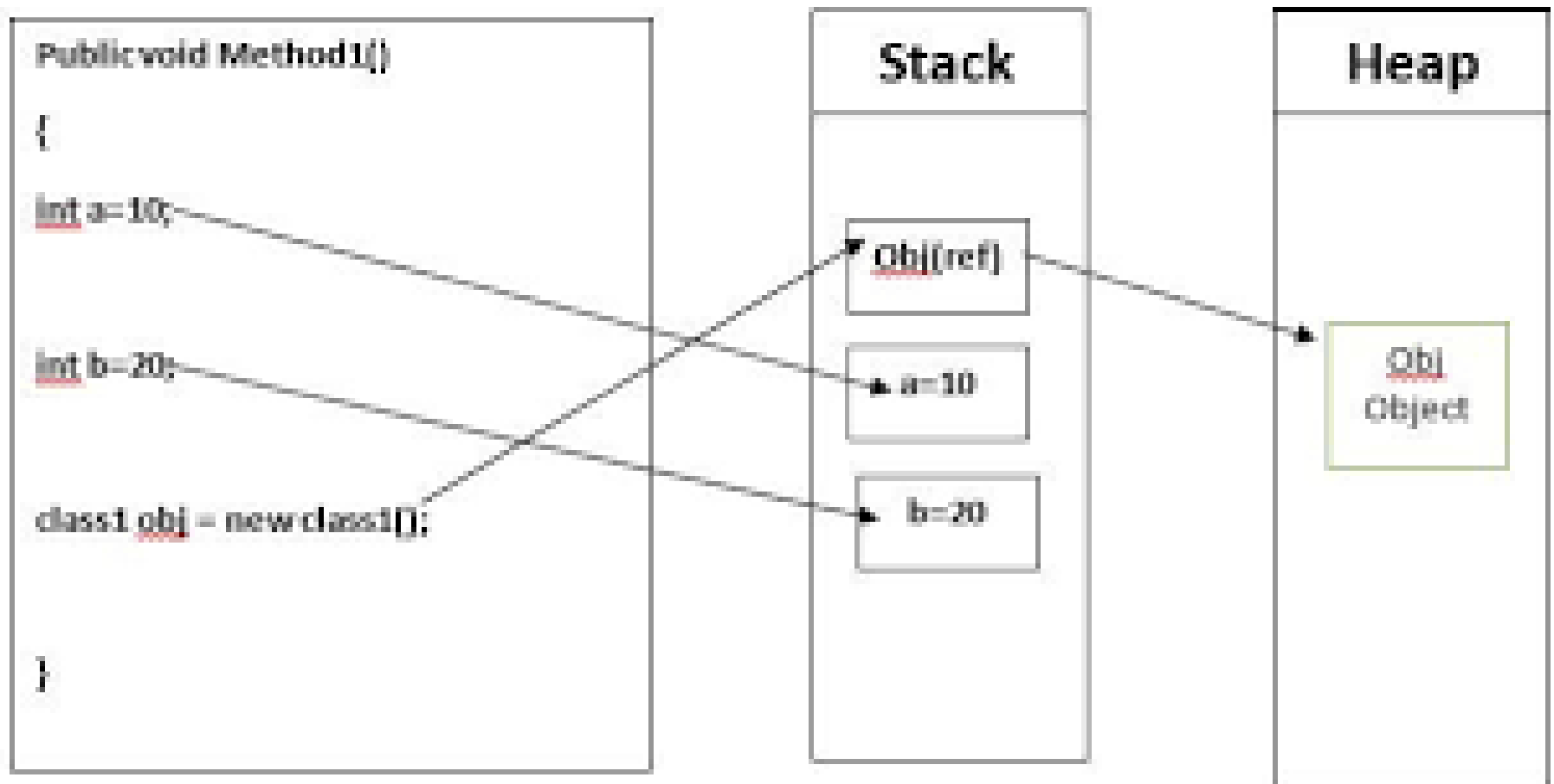
```
string s = "Hello World!!";
```



Reference type variable
contains address where the
value is stored

Actual value

Stack-Speicher <-> Heap-Speicher



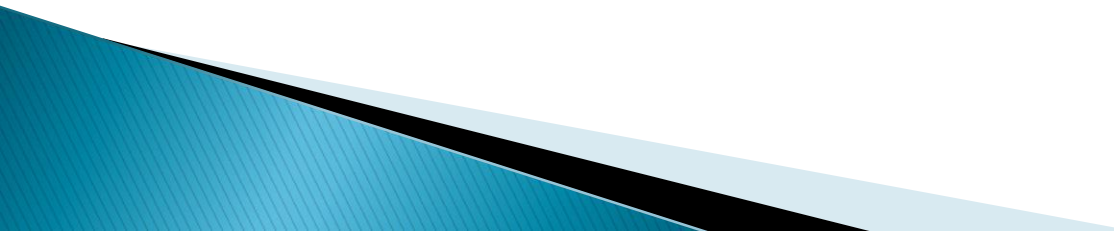
Stack-Speicher <-> Heap-Speicher

* *Stack*

- ▶ Begrenzte Größe
- ▶ LIFO Datenstruktur (die zuletzt angelegten Daten werden als erstes wieder freigegeben, deshalb auch "Stapel")
- ▶ Wächst und schrumpft mit dem Programmverlauf
- ▶ Wird verwendet für lokale Variablen und Funktionsparameter
- ▶ Kein explizites Freigeben des Speichers nötig
- ▶ Das Ablegen und Entfernen von Elementen ist sehr effizient

Stack-Speicher <-> Heap-Speicher

* *Heap*

- ▶ Der Heap kann innerhalb der Prozessgrenze beliebig groß werden
 - ▶ Anlegen und freigeben von Objekten ist vergleichsweise langsam
 - ▶ Auf dem Heap angelegte Objekte können global verfügbar gemacht werden
 - ▶ In Programmiersprachen ohne Garbage Collector muss der Speicher manuell freigegeben werden, wenn er nicht mehr benötigt wird
- 

Call by value <-> Call by reference

Beide Begriffe sind Arten der Parameterübergabe

* *Call by value*

- ▶ Wenn bei einem Aufruf einer Methode oder Funktion Variablen übergeben werden, wird diese als "Call by value" bezeichnet.
- ▶ Bei diesem Vorgang wird eine Kopie der Variable/n übergeben.

Call by value <-> Call by reference

Beide Begriffe sind Arten der Parameterübergabe

* *Call by reference*

- ▶ Wenn bei einem Aufruf einer Methode oder Funktion Adressen anstelle von Variablen übergeben werden, wird das aber als "Call by reference" bezeichnet.
- ▶ Bei diesem Vorgang wird die Adresse der Variable/n übergeben.

Quellen

- ▶ Quelle – DUDEN Informatik – ISBN 3-411-05233-3
 - ▶ Quelle – guru99.com
 - ▶ Quelle – lerneprogrammieren.com
 - ▶ Bild – wikipedia.com
 - ▶ Bild – net-informations.com
 - ▶ Bild – itwissen.info
 - ▶ Bild – pcynlitx.tech
 - ▶ Bild – codeproject.com
 - ▶ Bilder – turtorialsteacher.com
 - ▶ Bild – c-sharpcorner.com
- 