SAE	**Merkzettel**	Klasse:
L. Bung	**Polymorphie in Python**	Datum:

Problemsituation: Informationen und Fehlermeldungen werden gleich formatiert angezeigt.

```
System started
Disk not found
Connection established
Permission denied
```

Lösung: Wir schreiben zwei Klassen `InfoLog` und `ErrorLog`, die von der gemeinsamen Superklasse `LogEntry` erben. Die Formatierung (durch die Methode `display()`) wird durch die beiden Klassen unterschiedlich implementiert.

> **Polymorphie**
>
> Verschiedene Subklassen, die eine Methode von der Superklasse erben, können unterschiedliche Implementierungen der Methode haben. Damit ist es möglich, in verschiedenen Subklassen unterschiedliche Funktionalitäten bei Aufruf der Methode zu erreichen. Dieses Konzept nennt man in der objektorientierten Programmierung **Polymorphie**.

```python
class LogEntry:
  def __init__(self, message):
    self.message = message
  def display(self):
    print(self.message)

class InfoLog(LogEntry):
  def display(self):
    print(f"[INFO] {self.message}")

class ErrorLog(LogEntry):
  def display(self):
    print(f"[ERROR] {self.message} (!!!)")

log1 = InfoLog("System started")
log2 = ErrorLog("Disk not found")
log3 = InfoLog("Connection established")
log4 = ErrorLog("Permission denied")

log1.display()
log2.display()
log3.display()
log4.display()
```