LU05c - Generische Typen

In Java können Klassen und auch Interfaces einen oder gar mehrere generische Typen haben. Beispielsweise hat die Klasse`java.util.ArrayList<E>` als auch das implementierte Interface `java.util.List<E>` jeweils den generischen Typ „E“. Bei der Klasse `java.util.HashMap<K,V>` und dem Interface `Interface Map<K,V>` sind es die beiden generischen Typen „K“ und „V“.

Generische Typisierung ist praktisch, um den Entwickler bzw. die Entwicklerin zu leiten und die Absicht zur Verwendung der Klasse respektive des Interfaces darzulegen.

Der generische Typ wird bei einer Instanzierung mitgegeben:

```
var myList = new ArrayList<String>();
```

Der Diamond-Operator <> kann bei der instanzierten Klasse benutzt werden, wenn der generische Typ aus dem Klasse bzw. dem Interface abgeleitet werden kann.

```
List<String> myList = new ArrayList<>();
```

Der Vorteil von generischen Typen ist, dass die implementierten Methoden oder die übergebenen Attribute mit diesem Typ erzwungen werden können.

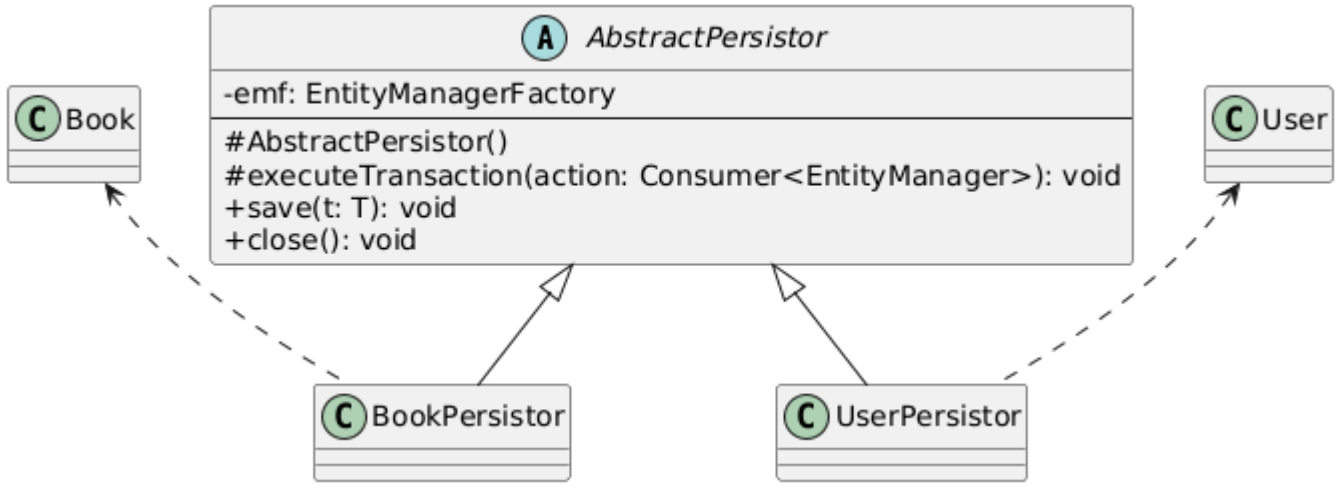
```
public interface List<E> ... {
    ...

    boolean add(E e);

    ...

    E get(int index);
    ...
}
```

```
public class ArrayList<E> ... implements List<E> ... {
    ...
    public boolean add(E e) {
        modCount++;
        add(e, elementData, size);
        return true;
    }
    ...
    public E get(int index) {
        Objects.checkIndex(index, size);
        return elementData(index);
    }
    ...
}
```

Generische Typen in der Library App

Eine Klasse oder ein Interface kann auch einen konkreten generischen Typ fix vererbt bekommen. In diesem Fall wird bei der Verwendung von `BookPersistor` immer `Book` als generischen Typ verwendet.

```
public class BookPersistor extends AbstractPersistor<Book> { ... }
```




```
public abstract class AbstractPersistor<T> implements AutoCloseable {

    protected final Logger log = LoggerFactory.getLogger(this.getClass());
    private final EntityManagerFactory emf;

    protected AbstractPersistor() {
        this.emf = Persistence.createEntityManagerFactory("localPU",
Config.getProperties());
    }

    public void save(T t) {
        executeTransaction(em -> em.merge(t));
    }

    protected void executeTransaction(Consumer<EntityManager> action) {
        try (EntityManager em = emf.createEntityManager()) {
            try {
                em.getTransaction().begin();
                action.accept(em);
                em.getTransaction().commit();
            } catch (RuntimeException e) {
                if (em.getTransaction().isActive()) {
                    em.getTransaction().rollback();
                }
                log.error("Error during transaction:", e);
                throw e;
            }
        }
```

```
    }

    @Override
    public void close() {
        if (emf != null && emf.isOpen()) {
            emf.close();
            log.info("EntityManagerFactory closed");
        }
    }
```

```
try (var bookPersistor = new BookPersistor()){
    bookPersistor.save(myBook);
}
```