|  |  |
| --- | --- |
| Beschreibung | Aktion |
| Leeres projekt | ./step 1 |
| Package erstellen | Cmd + N -> package eightyDays.java |
| Person implementieren | Cmd + N -> class Person  Felder name, firstname  Consturctor  Getter/Setter  Equals/HashCode |
| Bank implementieren | Cmd + N -> class Bank  Feld name  Consturctor  Getter/Setter  addPartner (beim hinzuefugen der gleichen person soll die gleiche UUID geliefert werden)  public UUID addPartner(Person pPerson) {  if (persons.containsValue(pPerson)) {  for(UUID id : persons.keySet()) {  if (persons.get(id).equals(pPerson)) {  return id;  }  }  }  UUID newId = UUID.*randomUUID*();  persons.put(newId, pPerson);  return newId; }  getPartner  public Person getPerson(UUID pId) {  return persons.get(pId); } |
| Tests ausfuehren | Tests vorbereteitet  ./step 2  Zeigen  Run Ctrl + Shift + R |
|  |  |
| Scala implementieren  Package erstellen | Package erstellen  Cmd + N -> package eightyDays.scala |
| Person implementieren | Cmd + N -> scala class Person  Case class  Felder name, firstname |
| Bank implementieren | Cmd + N -> scala class Bank  Feld val name  Feld var partners  **var** *persons* = *Map*[UUID, Person]()  Methode addPartner  **def** addPartner(pPerson: Person) = *persons* .find(\_.\_2 == pPerson)  .map(\_.\_1)  .getOrElse {  **val** newID = UUID.*randomUUID*()  *persons* = *persons* + (newID -> pPerson)  newID  }  Methode partner  **def** person(pId: UUID) = *persons* get pId |
| Tests | Tests zeigen: Person kein equals/hashCode, da dieser generiert ist  Tests ausfuehren |
| immutability | Scala favorisiert immutability. Map, name ist immutable  Wieso? |
| Slide Immutability |  |
| Java immutable machen | Person final und setter weg  Bank final und setter weg |
| Java functional machen | Bank addPartner umschreiben (Block mit Alt + Cursor markieren und mit Alt + Shift + Cursor bewegen)  public UUID addPartner(Person pPerson) {  return persons.entrySet().stream()  .filter(entry -> entry.getValue().equals(pPerson))  .findFirst()  .map(Map.Entry::getKey)  .orElseGet(() -> {  UUID newID = UUID.*randomUUID*();  persons.put(newID, pPerson);  return newID;  }); }  Bank getPartner mit Optional (Consumer soll sich nicht mit null befassen, sondern es wird immer etwas geliefert. Ist Null-Object-Pattern)  public Optional<Person> getPerson(UUID pId) {  return Optional.*ofNullable*(persons.get(pId)); } |
| Tests ausfuehren | Sind noch gruen. Nichts gebrochen (./step 4 definiert den Stand) |