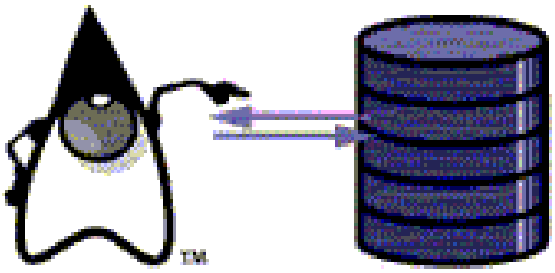
JDBC TAKEN

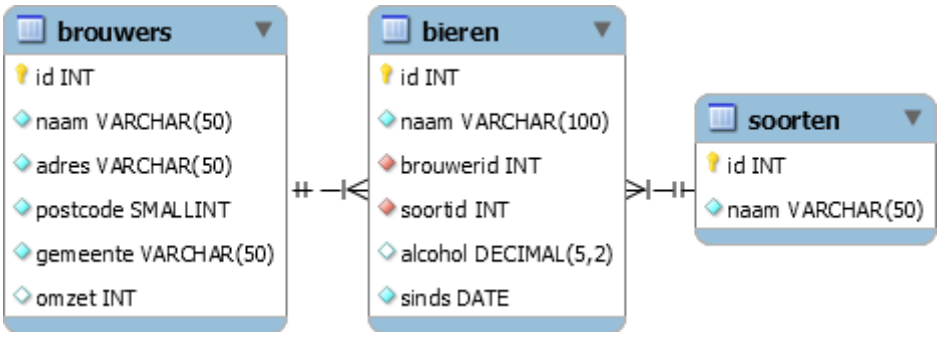


INHOUD

1 Taken

1.1 Repository

Gebruik de database `bieren`. Maak die met het script `bieren.sql`.



Maak in IntelliJ een apart project voor de taken.
Maak een class AbstractRepository voor de database bieren. Gebruik de username `cursist`.

1.2 Bieren verwijderen

Maak een class BierRepository.
Maak daarin een method die alle bieren verwijdert waarvan het alcohol% niet ingevuld is.
Roep de method op vanuit een class Main.

1.3 Gemiddelde

Toon de gemiddelde omzet van alle brouwers.
Toon daarna een lijst met de brouwers die een omzet hebben die hoger is dan de gemiddelde omzet.

1.4 Van tot

De gebruiker typt een minimum en een maximum omzet.
Toon een lijst met de brouwers waarvan de omzet ligt tussen dit minimum en maximum.
Sorteer op omzet. Sorteer brouwers met dezelfde omzet op id.

1.5 Id

De gebruiker typt de id van een brouwer.
Als je de brouwer vindt, toon je van de brouwer de id, de naam, de gemeente en de omzet.
Als je de brouwer niet vindt, toon je de tekst “Brouwer niet gevonden.”.

1.6 Stored procedure

Maak dezelfde taak als de taak “Van tot”, maar gebruik deze keer een stored procedure.

1.7 Failliet

Brouwer `1` gaat failliet.
Zijn bieren met een alcohol% vanaf `8.5` worden overgenomen door brouwer `2`.
Zijn andere bieren worden overgenomen door brouwer `3`.
Verwijder daarna brouwer `1`.
Doe al deze bewerkingen binnen één transactie.

1.8 Isolation

Gebruik in de taak “Failliet” het optimaalste transaction isolation level.

1.9 Bieren van een maand

De gebruiker typt een maandnummer (getal tussen 1 en 12).
Toon de bieren die voor het eerst verkocht zijn in die maand (over alle jaren heen).
Toon per bier de naam. Sorteert op naam.

1.10 Aantal bieren per brouwer

Toon het aantal bieren per brouwer. Toon per brouwer de naam en zijn aantal bieren.
Sorteer op de namen van de brouwers.

1.11 Bieren van een soort

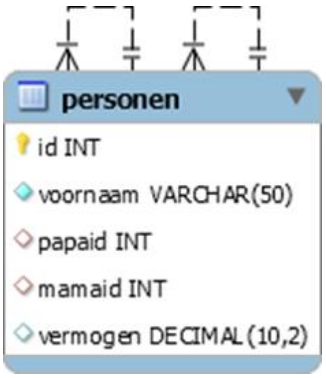
De gebruiker typt de naam van een soort. Toon de namen van de bieren van die soort.

1.12 Omzet leegmaken

De gebruiker typt brouwer nummers, tot hij `0` typt.
Als hij daarbij een negatief nummer typt, toon je een foutmelding.
Als hij daarbij een nummer typt dat hij reeds typte, toon je een foutmelding.
Maak de omzet van alle geselecteerde brouwers leeg.
Als de gebruiker nummers typte die niet bestaan in de database, toon je deze nummers.

1.13 Gezin

Gebruik de database `familie`. Maak die met het script `familie.sql`.



De tabel `personen` heeft twee relaties met zichzelf:

- `papaid` is een foreign key die verwijst naar de primary key `id`. `papaid` bevat de `id` in van de papa van de huidige persoon.
- `mamaid` is een foreign key die verwijst naar de primary key `id`. `mamaid` bevat de `id` in van de mama van de huidige persoon.

Voorbeeld: papa `Hans`,mama `Alexandra`
en hun kinderen `Aeneas`, `Alissia` en `Anaïs`:

id	voornaam	papaid	mamaid	vermogen
1	Hans			900
2	Alexandra			600
3	Aeneas	1	2	0
4	Alissia	1	2	0
5	Anaïs	1	2	0

De gebruiker typt de voornaam van een papa en de voornaam van een mama.
De gebruiker typt daarna de voornamen van hun kinderen, tot hij `STOP` typt.
Voeg dit gezin toe als records in de tabel `personen`.

1.14 Erfenis

Als een persoon overlijdt erven zijn kinderen zijn vermogen. Als de persoon 3 kinderen heeft, erft elk kind 1/3 van het vermogen van die persoon. De gebruiker typt de id van de overledene.
Verdeel zijn vermogen over zijn kinderen (als hij kinderen heeft).
Wijzig daarna het vermogen van de persoon naar nul.

2 Voorbeeldoplossingen

2.1 Repository

2.1.1 AbstractRepository

```
package be.vdab.repositories;
// enkele imports uit de package java.sql
abstract class AbstractRepository {
  private static final String URL = "jdbc:mysql://localhost/bieren";
  private static final String USER = "cursist";
  private static final String PASSWORD = "cursist";
  protected Connection getConnection() throws SQLException {
    return DriverManager.getConnection(URL, USER, PASSWORD);
  }
}
```

2.2 Bieren verwijderen

2.2.1 BierRepository

```
package be.vdab.repositories;
public class BierRepository extends AbstractRepository {
  public int verwijderBierenMetOnbekendeAlcohol() throws SQLException {
    try (var connection = super.getConnection();
      var statement = connection.prepareStatement(
        "delete from bieren where alcohol is null")) {
      return statement.executeUpdate();
    }
  }
}
```

2.2.2 Main: code in de method main

```
var repository = new BierRepository();
try {
  System.out.print(repository.verwijderBierenMetOnbekendeAlcohol());
  System.out.println(" bieren verwijderd.");
} catch (SQLException ex) {
  ex.printStackTrace(System.err);
}
```

2.3 Gemiddelde

2.3.1 Brouwer

```
package be.vdab.domain;
public class Brouwer {
  private final long id;
  private final String naam;
  private final String adres;
  private final int postcode;
  private final String gemeente;
  private final int omzet;
  // constructor met parameters
  @Override
  public String toString() {
    return id + ":" + naam + " (" + gemeente + ") " + omzet;
  }
}
```

2.3.2 BrouwerRepository

```
package be.vdab.repositories;
// enkele imports
public class BrouwerRepository extends AbstractRepository {
  public BigDecimal findGemiddeldeOmzet() throws SQLException {
    try (var connection = super.getConnection();
      var statement = connection.prepareStatement(
        "select avg(omzet) as gemiddelde from brouwers")) {
      var result = statement.executeQuery();
      result.next();
      return result.getBigDecimal("gemiddelde");
    }
  }
  private Brouwer naarBrouwer(ResultSet result) throws SQLException {
    return new Brouwer(result.getLong("id"), result.getString("naam"),
      result.getString("adres"), result.getInt("postcode"),
      result.getString("gemeente"), result.getInt("omzet"));
  }
  public List<Brouwer> findByOmzetGroterDanGemiddelde() throws SQLException {
    var sql = "select id,naam,adres,postcode,gemeente,omzet from brouwers" +
      " where omzet > (select avg(omzet) from brouwers)";
    try (var connection = super.getConnection();
      var statement = connection.prepareStatement(sql) {
      var brouwers = new ArrayList<Brouwer>();
      var result = statement.executeQuery();
      while (result.next()) {
        brouwers.add(naarBrouwer(result));
      }
      return brouwers;
    }
  }
}
```

2.3.3 Main: code in de method main

```
var repository = new BrouwerRepository();
try {
  System.out.println(repository.findGemiddeldeOmzet());
  repository.findByOmzetGroterDanGemiddelde().forEach(System.out::println);
} catch (SQLException ex) {
  ex.printStackTrace(System.err);
}
```

2.4 Van tot

2.4.1 BrouwerRepository: extra method

```
public List<Brouwer> findByOmzetTussen(int minimum, int maximum) throws SQLException {
  var sql = "select id,naam,adres,postcode,gemeente,omzet from brouwers" +
    " where omzet between ? and ? order by omzet,id";
  try (var connection = super.getConnection();
    var statement = connection.prepareStatement(sql)) {
    statement.setInt(1, minimum);
    statement.setInt(2, maximum);
    var brouwers = new ArrayList<Brouwer>();
    var result = statement.executeQuery();
    while (result.next()) {
      brouwers.add(naarBrouwer(result));
    }
    return brouwers;
  }
}
```

2.4.2 Main: code in de method main

```
var scanner = new Scanner(System.in);
System.out.print("Minimum:");
var minimum = scanner.nextInt();
System.out.print("Maximum:");
var maximum = scanner.nextInt();
var repository = new BrouwerRepository();
try {
  repository.findByOmzetTussen(minimum, maximum).forEach(System.out::println);
} catch (SQLException ex) {
  ex.printStackTrace(System.err);
}
```

2.5 Id

2.5.1 BrouwerRepository: extra method

```
public Optional<Brouwer> findById(long id) throws SQLException {
  var sql = "select id,naam,adres,postcode,gemeente,omzet from brouwers where id=?";
  try (var connection = super.getConnection();
    var statement = connection.prepareStatement(sql)) {
    statement.setLong(1, id);
    var result = statement.executeQuery();
    return result.next() ? Optional.of(naarBrouwer(result)) : Optional.empty();
  }
}
```

2.5.2 Main: code in de method main

```
System.out.print("id:");
var scanner = new Scanner(System.in);
var id = scanner.nextLong();
var repository = new BrouwerRepository();
try {
  repository.findById(id)
    .ifPresentOrElse(System.out::println),
       () -> System.out.println("Niet gevonden"));
} catch (SQLException ex) {
  ex.printStackTrace(System.err);
}
```

2.6 Stored procedure

2.6.1 Stored procedure

```
create procedure bieren.BrouwersMetOmzetTussen(minimum int, maximum int)
begin
select id, naam, adres, postcode, gemeente, omzet
from brouwers
where omzet between minimum and maximum
order by omzet, id;
end
```

2.6.2 Rechten

```
grant execute on procedure bieren.BrouwersMetOmzetTussen to cursist
```

2.6.3 BrouwerRepository: gewijzigde method findByOmzetTussen:

```
public List<Brouwer> findByOmzetTussen(int van, int tot) throws SQLException {
  var call = "{call BrouwersMetOmzetTussen(?,?)}";
  try (var connection = super.getConnection();
    var statement = connection.prepareCall(call)) {
    statement.setInt(1, van);
    statement.setInt(2, tot);
    var brouwers = new ArrayList<Brouwer>();
    var result = statement.executeQuery();
    while (result.next()) {
      brouwers.add(naarBrouwer(result));
    }
    return brouwers;
  }
}
```

2.7 Failliet

2.7.1 BrouwerRepository: extra method:

```
public void brouwer1Failliet() throws SQLException {
  var sqlNaarBrouwer2 =
    "update bieren set brouwerid=2 where brouwerid=1 and alcohol>=8.5";
  var sqlNaarBrouwer3 = "update bieren set brouwerid = 3 where brouwerid = 1";
  var sqlDeleteBrouwer1 = "delete from brouwers where id = 1";
  try (var connection = super.getConnection();
    var statementNaarBrouwer2 = connection.prepareStatement(sqlNaarBrouwer2);
    var statementNaarBrouwer3 = connection.prepareStatement(sqlNaarBrouwer3);
    var statementDeleteBrouwer1 = connection.prepareStatement(sqlDeleteBrouwer1)){
    connection.setAutoCommit(false);
    statementNaarBrouwer2.executeUpdate();
    statementNaarBrouwer3.executeUpdate();
    statementDeleteBrouwer1.executeUpdate();
    connection.commit();
  }
}
```

2.7.2 Main: code in de method main:

```
var repository = new BrouwerRepository();
try {
  repository.brouwer1Failliet();
  } catch (SQLException ex) {
  ex.printStackTrace(System.err);
}
```

2.8 Isolation

2.8.1 BrouwerRepository

Extra opdracht in de method brouwer1Failliet, voor de opdracht connection.setAutoCommit(false);

```
connection.setTransactionIsolation(Connection.TRANSACTION_READ_COMMITTED);
```

2.9 Bieren van een maand

2.9.1 BierRepository: extra method:

```
public List<String> findByMaand(int maand) throws SQLException {
  var sql = "select naam from bieren where {fn month(sinds)} = ? order by naam";
  try (var connection = super.getConnection();
    var statement = connection.prepareStatement(sql)) {
    connection.setTransactionIsolation(Connection.TRANSACTION_READ_COMMITTED);
    connection.setAutoCommit(false);
```

```
    statement.setInt(1, maand);
    var namen = new ArrayList<String>();
    var result = statement.executeQuery();
    while (result.next()) {
      namen.add(result.getString("naam"));
    }
    connection.commit();
    return namen;
  }
}
```

2.9.2 Main: code in de method main:

```
var scanner = new Scanner(System.in);
System.out.print("Maand:");
var maand = scanner.nextInt();
while (maand < 1 || maand > 12) {
  System.out.println("Verkeerd, maand:");
  maand = scanner.nextInt();
}
var repository = new BierRepository();
try {
  repository.findByMaand(maand).forEach(System.out::println);
} catch (SQLException ex) {
  ex.printStackTrace(System.err);
}
```

2.10 Aantal bieren per brouwer

2.10.1 BrouwerNaamEnAantalBieren

```
package be.vdab.dto;
public class BrouwerNaamEnAantalBieren {
  private final String naam;
  private final int aantalBieren;
  // constructor met parameters, toString die naam en - en aantalBieren teruggeeft
}
```

2.10.2 BrouwerRepository: extra method:

```
public List<BrouwerNaamEnAantalBieren> findBrouwerNamenEnAantalBieren()
  throws SQLException {
  var sql = "select brouwers.naam,count(*) as aantal from brouwers " +
    "inner join bieren on brouwers.id=bieren.brouwerid group by brouwers.naam " +
    "order by brouwers.naam";
  try (var connection = super.getConnection();
    var statement = connection.prepareStatement(sql)) {
    connection.setTransactionIsolation(Connection.TRANSACTION_READ_COMMITTED);
    connection.setAutoCommit(false);
    var list = new ArrayList< BrouwerNaamEnAantalBieren >();
    var result = statement.executeQuery();
    while (result.next()) {
      list.add(new BrouwerNaamEnAantalBieren(
        result.getString("naam"), result.getInt("aantal")));
    }
    connection.commit();
    return list;
  }
}
```

2.10.3 Main: code in de method main:

```
var repository = new BrouwerRepository();
try {
  repository.findBrouwerNamenEnAantalBieren().forEach(System.out::println);
} catch (SQLException ex) {
  ex.printStackTrace(System.err);
}
```

2.11 Bieren van een soort

2.11.1 BierRepository: extra method

```
public List<String> findBySoort(String soort) throws SQLException {
  var sql = "select naam from bieren where soortid=(select id from soorten where naam=?)";
  try (var connection = super.getConnection();
    var statement = connection.prepareStatement(sql)) {
    connection.setTransactionIsolation(Connection.TRANSACTION_READ_COMMITTED);
    connection.setAutoCommit(false);
    statement.setString(1, soort);
    var namen = new ArrayList<String>();
    var result = statement.executeQuery();
    while (result.next()) {
      namen.add(result.getString("naam"));
    }
    connection.commit();
    return namen;
  }
}
```

2.11.2 Main: code in de method main

```
System.out.print("soort:");
var scanner = new Scanner(System.in);
var soort = scanner.nextLine();
var repository = new BierRepository();
try {
  var namen = repository.findBySoort(soort);
  if (namen.isEmpty()) {
    System.out.println("Geen bieren gevonden");
  } else {
    namen.forEach(System.out::println);
  }
} catch (SQLException ex) {
  ex.printStackTrace(System.err);
}
```

2.12 Omzet leegmaken

2.12.1 BrouwerRepository: extra methods

```
public int maakOmzetLeeg(Set<Long> ids) throws SQLException {
  if (ids.isEmpty()) {
    return 0;
  }
  var sql = "update brouwers set omzet=null where id in (" +
    "?,".repeat(ids.size()- 1)) + "?)";
  try (var connection = super.getConnection();
    var statement = connection.prepareStatement(sql)) {
    connection.setTransactionIsolation(Connection.TRANSACTION_READ_COMMITTED);
    connection.setAutoCommit(false);
```

```
    int index = 1;
    for (var id : ids) {
      statement.setLong(index++, id);
    }
    var aantalAangepast = statement.executeUpdate();
    connection.commit();
    return aantalAangepast;
  }
}
public Set<Long> findIds(Set<Long> ids) throws SQLException {
  if (ids.isEmpty()) {
    return Set.of();
  }
  var sql = "select id from brouwers where id in ("
    + "?,".repeat(ids.size()- 1)) + "?)";
  try (var connection = super.getConnection();
    var statement = connection.prepareStatement(sql)) {
    connection.setTransactionIsolation(Connection.TRANSACTION_READ_COMMITTED);
    connection.setAutoCommit(false);
    var index = 1;
    for (var id : ids) {
      statement.setLong(index++, id);
    }
    var gevondenIds = new HashSet<Long>();
    var result = statement.executeQuery();
    while (result.next()) {
      gevondenIds.add(result.getLong("id"));
    }
    return gevondenIds;
  }
}
```

2.13 Main

2.13.1 Extra method

```
private static Set<Long> vraagIds() {
  var ids = new LinkedHashSet<Long>();
  var scanner = new Scanner(System.in);
  System.out.print("id (stop met 0):");
  for (long id; (id = scanner.nextLong()) != 0;) {
    System.out.print("id (stop met 0):");
    if (id < 0) {
      System.out.print("nummer moet positief zijn, probeer opnieuw:");
    } else {
      if ( ! ids.add(id)) {
        System.out.print(id + " reeds getypt, probeer opnieuw:");
      }
    }
  }
  return ids;
}
```

2.13.2 Code in de method main

```
var ids = vraagIds();
if (!ids.isEmpty()) {
  var repository = new BrouwerRepository();
  try {
    if (repository.maakOmzetLeeg(ids) != ids.size()) {
      System.out.println("Niet gevonden ids:");
      var gevondenIds = repository.findIds(ids);
      ids.stream().filter(id -> ! gevondenIds.contains(id))
        .forEach(System.out::println);
    }
  } catch (SQLException ex) {
    ex.printStackTrace(System.err);
  }
}
```

2.14 Gezin

2.14.1 AbstractRepository

```
package be.vdab.repositories;
// enkele imports
abstract class AbstractRepository {
  private static final String URL = "jdbc:mysql://localhost/familie";
  private static final String USER = "cursist";
  private static final String PASSWORD = "cursist";
  protected Connection getConnection() throws SQLException {
    return DriverManager.getConnection(URL, USER, PASSWORD);
  }
}
```

2.14.2 Gezin

```
package be.vdab.domain;
import java.util.ArrayList;
import java.util.List;
// enkele imports
public class Gezin {
  private final String papa;
  private final String mama;
  private final List<String> kinderen = new ArrayList<>();
  public Gezin(String papa, String mama) {
    this.papa = papa;
    this.mama = mama;
  }
  public void addKind(String kind) {
    kinderen.add(kind);
  }
  // getters voor papa, mama en kinderen
}
```

2.14.3 PersoonRepository

```
package be.vdab.repositories;
// enkele imports
public class PersoonRepository extends AbstractRepository {
  public void create(Gezin gezin) throws SQLException {
    var insertOuder = "insert into personen(voornaam) values (?)";
    var insertKind = "insert into personen(voornaam,papaid,mamaid) values (?,?,?)";
    try (var connection = super.getConnection();
      var statementOuder = connection.prepareStatement(insertOuder,
        PreparedStatement.RETURN_GENERATED_KEYS)) {
```

```
      connection.setTransactionIsolation(Connection.TRANSACTION_READ_COMMITTED);
      connection.setAutoCommit(false);
      statementOuder.setString(1, gezin.getPapa());
      statementOuder.addBatch();
      statementOuder.setString(1, gezin.getMama());
      statementOuder.addBatch();
      statementOuder.executeBatch();
      var result = statementOuder.getGeneratedKeys();
      result.next();
      var papaId = result.getLong(1);
      result.next();
      var mamaId = result.getLong(1);
      try (var statementKind = connection.prepareStatement(insertKind)) {
        statementKind.setLong(2, papaId);
        statementKind.setLong(3, mamaId);
        for (String kind : gezin.getKinderen()) {
          statementKind.setString(1, kind);
          statementKind.addBatch();
        }
        statementKind.executeBatch();
      }
      connection.commit();
    }
  }
}
```

2.14.4 Main

```
package be.vdab;
// enkele imports
class Main {
  public static void main(String[] args) {
    var scanner = new Scanner(System.in);
    System.out.print("Papa:");
    var papa = scanner.nextLine();
    System.out.print("Mama:");
    var mama = scanner.nextLine();
    var gezin = new Gezin(papa, mama);
    System.out.println("Kinderen (typ STOP om te stoppen):");
    for (String kind; ! "STOP".equals((kind = scanner.nextLine()));) {
      gezin.addKind(kind);
    }
    try {
      var repository = new PersoonRepository();
      repository.create(gezin);
    } catch (SQLException ex) {
      ex.printStackTrace(System.err);
    }
  }
}
```

2.15 Erfenis

2.15.1 PersoonNietGevondenException

```
package be.vdab.exceptions;
public class PersoonNietGevondenException extends RuntimeException {
  private static final long serialVersionUID = 1L;
}
```

2.15.2 PersoonRepository: extra methods

```
public void erfenis(long id) throws SQLException {
  try (var connection = super.getConnection()) {
    connection.setTransactionIsolation(Connection.TRANSACTION_READ_COMMITTED);
    connection.setAutoCommit(false);
    var optionalVermogen = findVermogen(id, connection);
    if (optionalVermogen.isPresent()) {
      var vermogen = optionalVermogen.get();
      if (vermogen.compareTo(BigDecimal.ZERO) > 0) {
        var aantalKinderen = findAantalKinderen(id, connection);
        if (aantalKinderen != 0) {
          var erfenisPerKind = vermogen.divide(
            BigDecimal.valueOf(aantalKinderen), 2, RoundingMode.HALF_UP);
            verHoogVermogenMetErfenis(id, connection, erfenisPerKind);
        }
        zetVermogenOpNul(id, connection);
      }
      connection.commit();
      return;
    }
    connection.rollback();
    throw new PersoonNietGevondenException();
  }
}
private Optional<BigDecimal> findVermogen(long id, Connection connection)
  throws SQLException {
  try (var statement = connection.prepareStatement(
    "select vermogen from personen where id = ?")) {
    statement.setLong(1, id);
    var result = statement.executeQuery();
    if (result.next()) {
      return Optional.of(result.getBigDecimal("vermogen"));
    }
    return Optional.empty();
  }
}
private void zetVermogenOpNul(long id,Connection connection) throws SQLException {
  try (var statement = connection.prepareStatement(
    "update personen set vermogen= 0 where id = ?")) {
    statement.setLong(1, id);
    statement.executeUpdate();
  }
}
private int findAantalKinderen(long id,Connection connection) throws SQLException{
  var sql =
    "select count(*) as aantalKinderen from personen where papaid=? or mamaid=?";
  try (var statement = connection.prepareStatement(sql)) {
    statement.setLong(1, id);
    statement.setLong(2, id);
    var result = statement.executeQuery();
    result.next();
    return result.getInt("aantalKinderen");
  }
}
```

```
private void verHoogVermogenMetErfenis(long id, Connection connection,
  BigDecimal erfenis) throws SQLException {
  var sql =
    "update personen set vermogen = vermogen+? where papaid=? or mamaid=?";
  try (var statement = connection.prepareStatement(sql)) {
    statement.setBigDecimal(1, erfenis);
    statement.setLong(2, id);
    statement.setLong(3, id);
    statement.executeUpdate();
  }
}
```

2.15.3 Main: code in de method main

```
var scanner = new Scanner(System.in);
System.out.print("Id:");
var id = scanner.nextLong();
try {
  var repository = new PersoonRepository();
  repository.erfenis(id);
} catch (PersoonNietGevondenException ex) {
  System.out.println("Persoon niet gevonden");
} catch (SQLException ex) {
  ex.printStackTrace(System.err);
}
```

COLOFON

Domeinexpertisemanager	Jean Smits
Moduleverantwoordelijke	
Auteurs	Hans Desmet
Versie	6/1/2021
Codes	Peoplesoftcode: Wettelijk depot:

Omschrijving module-inhoud

Abstract	Doelgroep	Opleiding Java Ontwikkelaar
	Aanpak	Zelfstudie
	Doelstelling	JDBC kunnen gebruiken
Trefwoorden		JDBC
Bronnen/meer info		