Programmiergrundlagen

Datenbanktechnologien

Prof. Dr. Ingo Claßen

Hochschule für Technik und Wirtschaft Berlin

Eclipse Workspace anlegen

Projekte herunterladen

Projekte importieren

Debugging

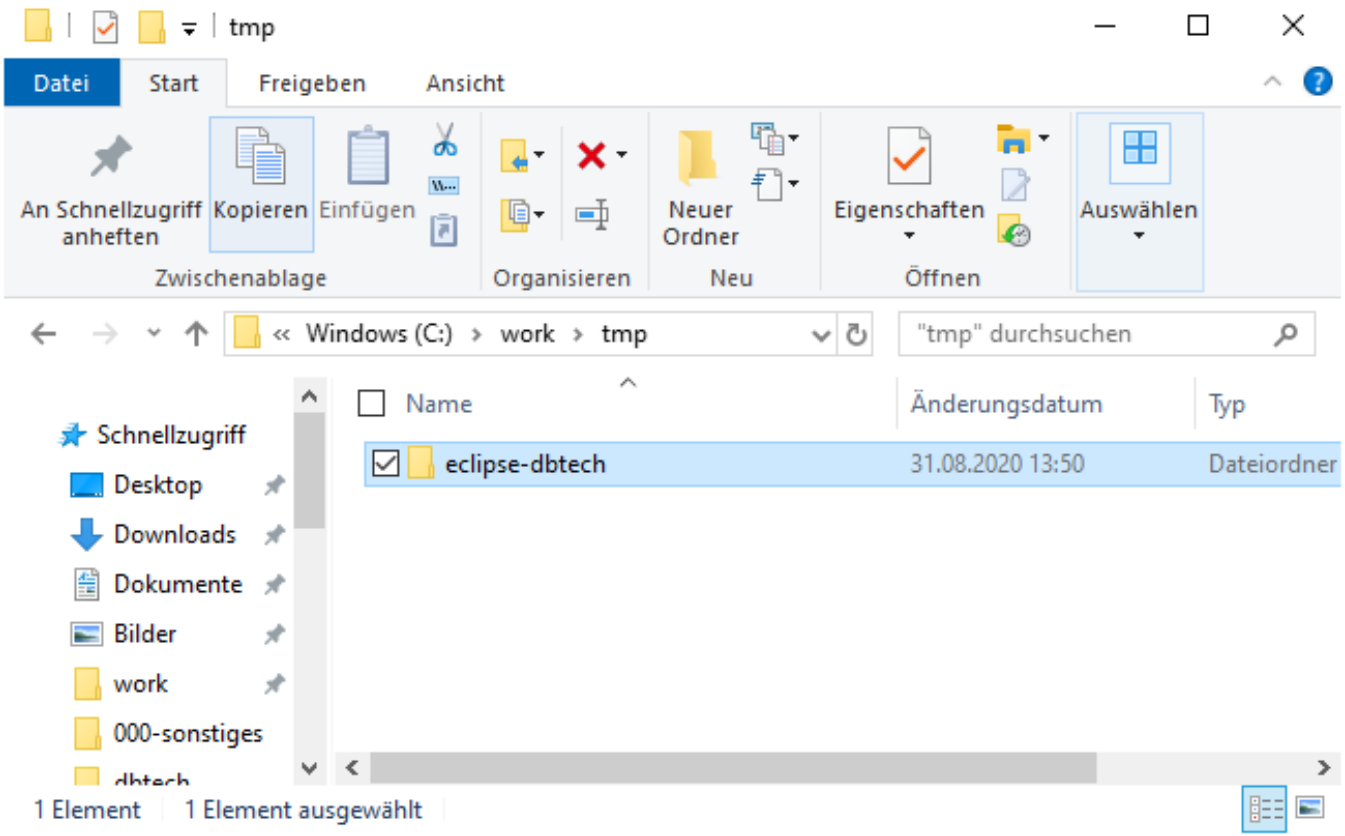
Logging

Exceptions

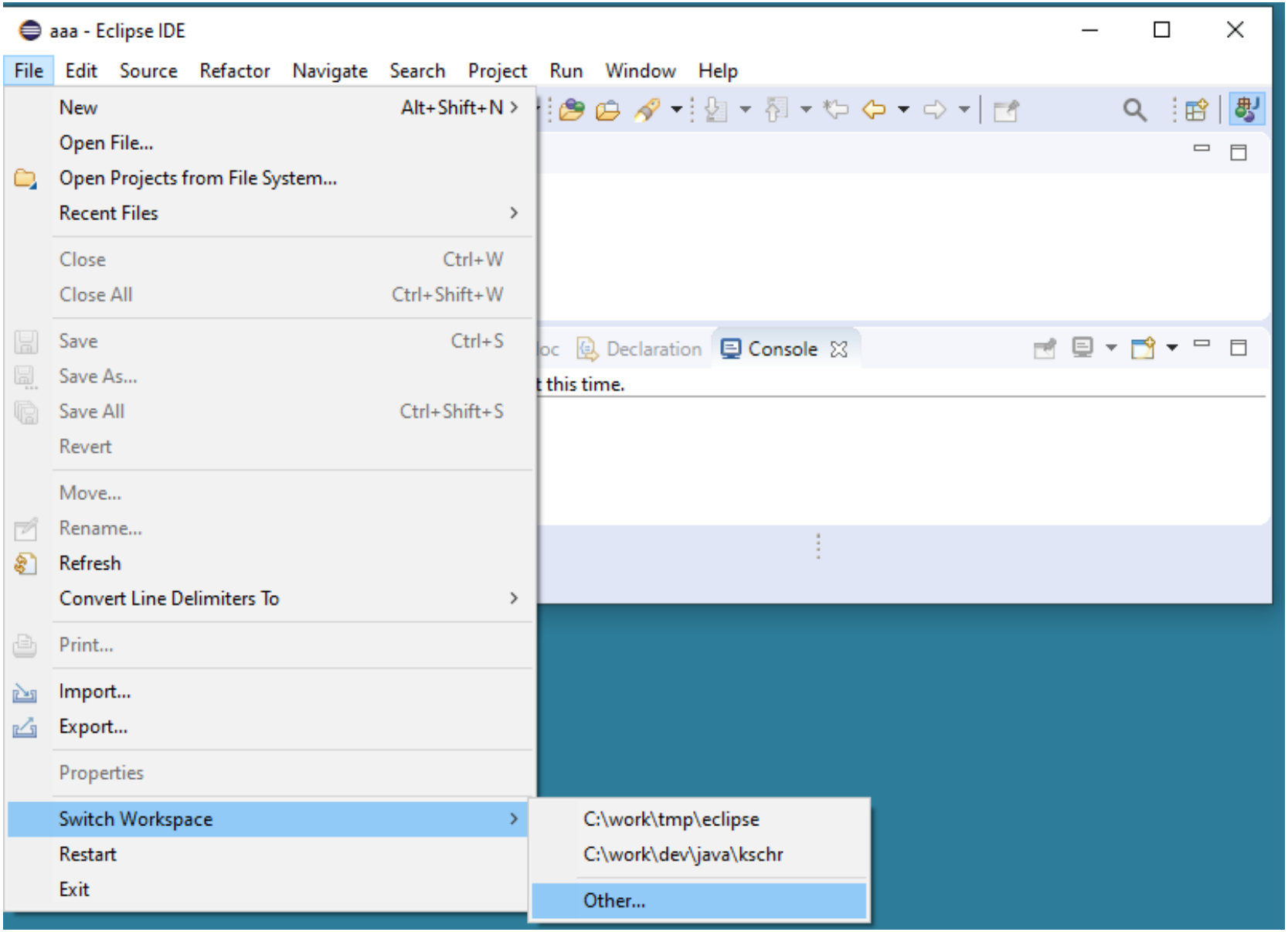
Programmstruktur – Dienste

Verzeichnis für neuen Workspace anlegen

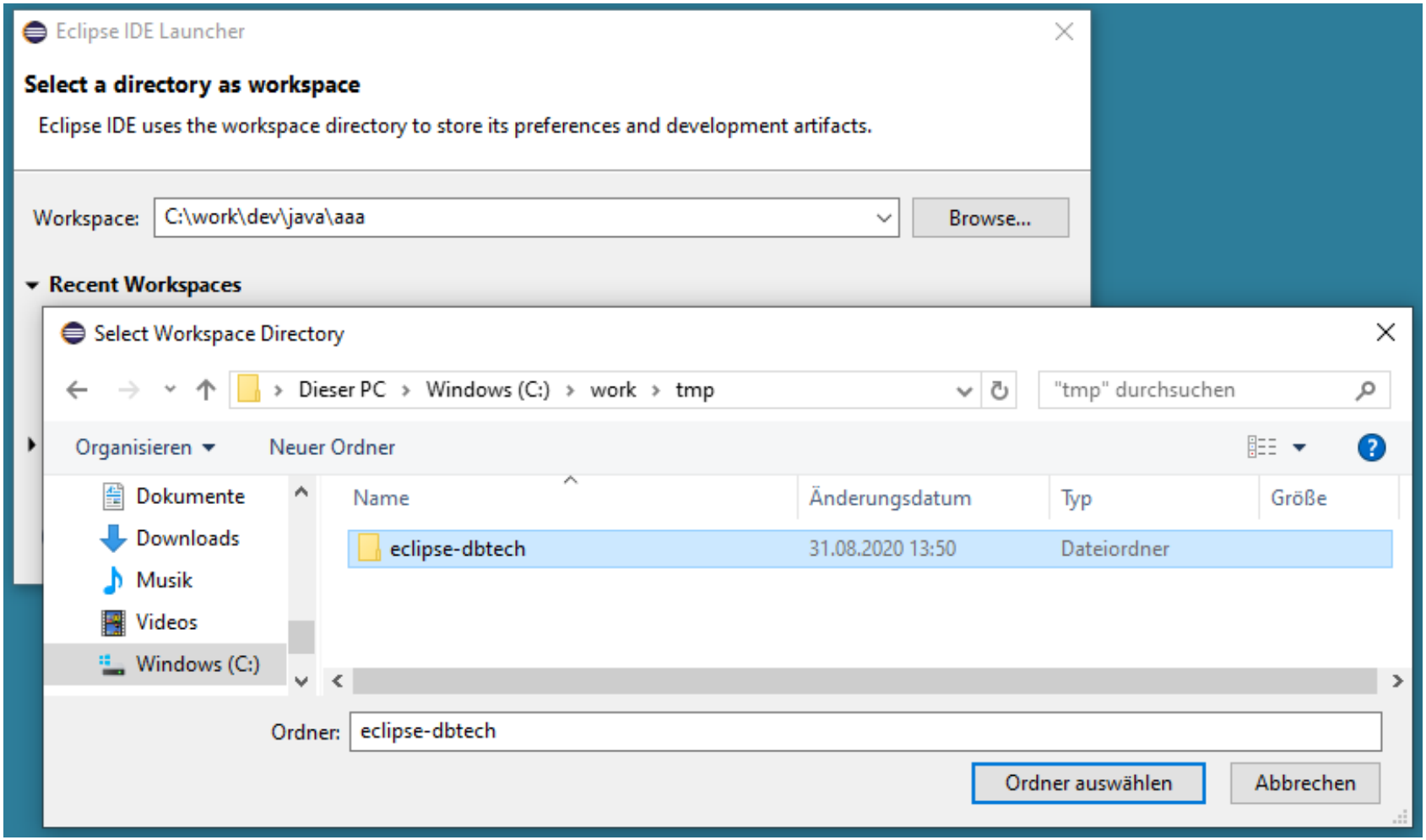
- Legen Sie an geeigneter Stelle auf Ihrem Computer ein Verzeichnis für den neuen Workspace an
- Als Beispiel verwende ich `eclipse-dbtech` an der Stelle `C:/work/tmp`



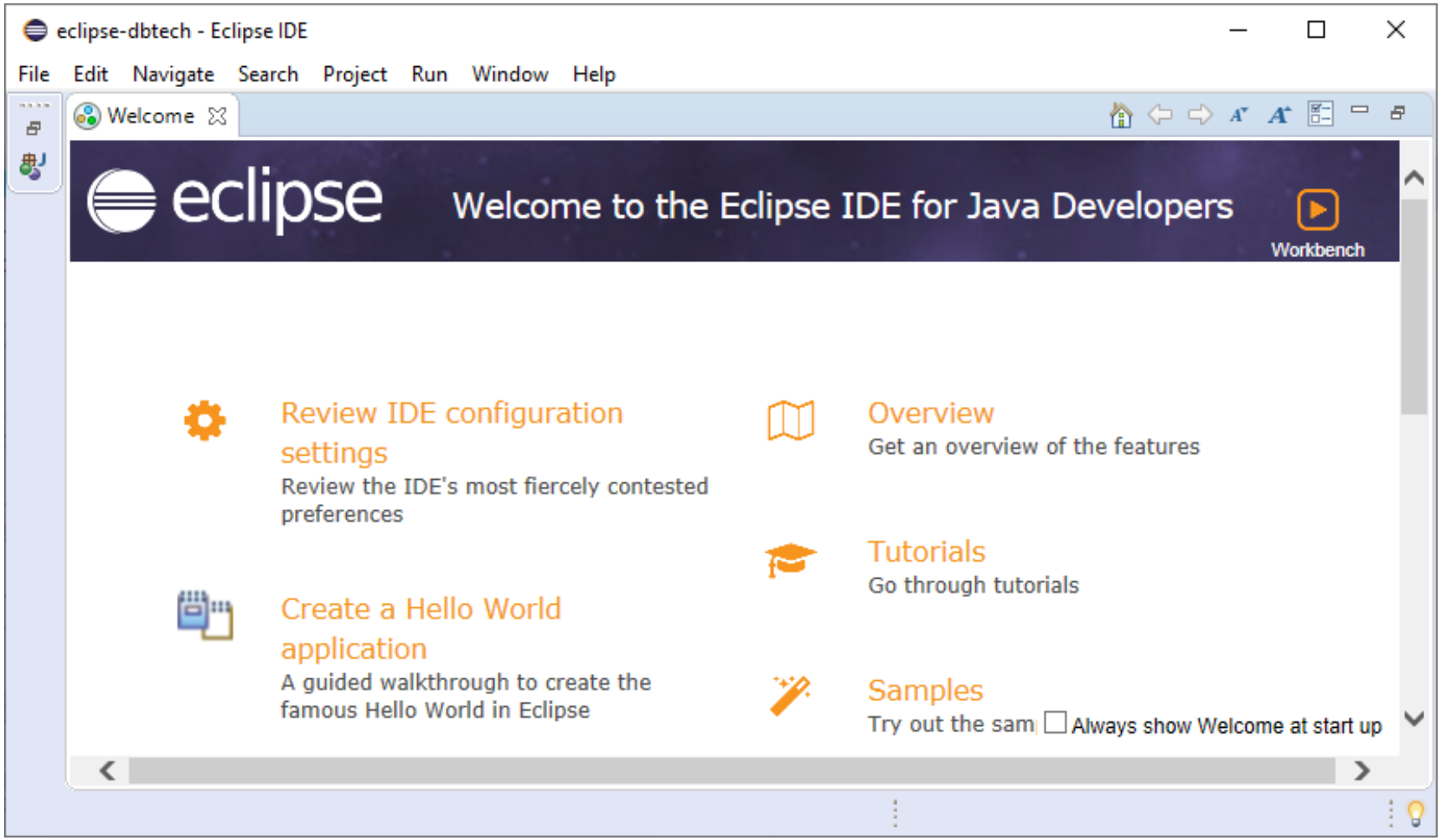
Auf neuen Workspace wechseln



Verzeichnis auswählen

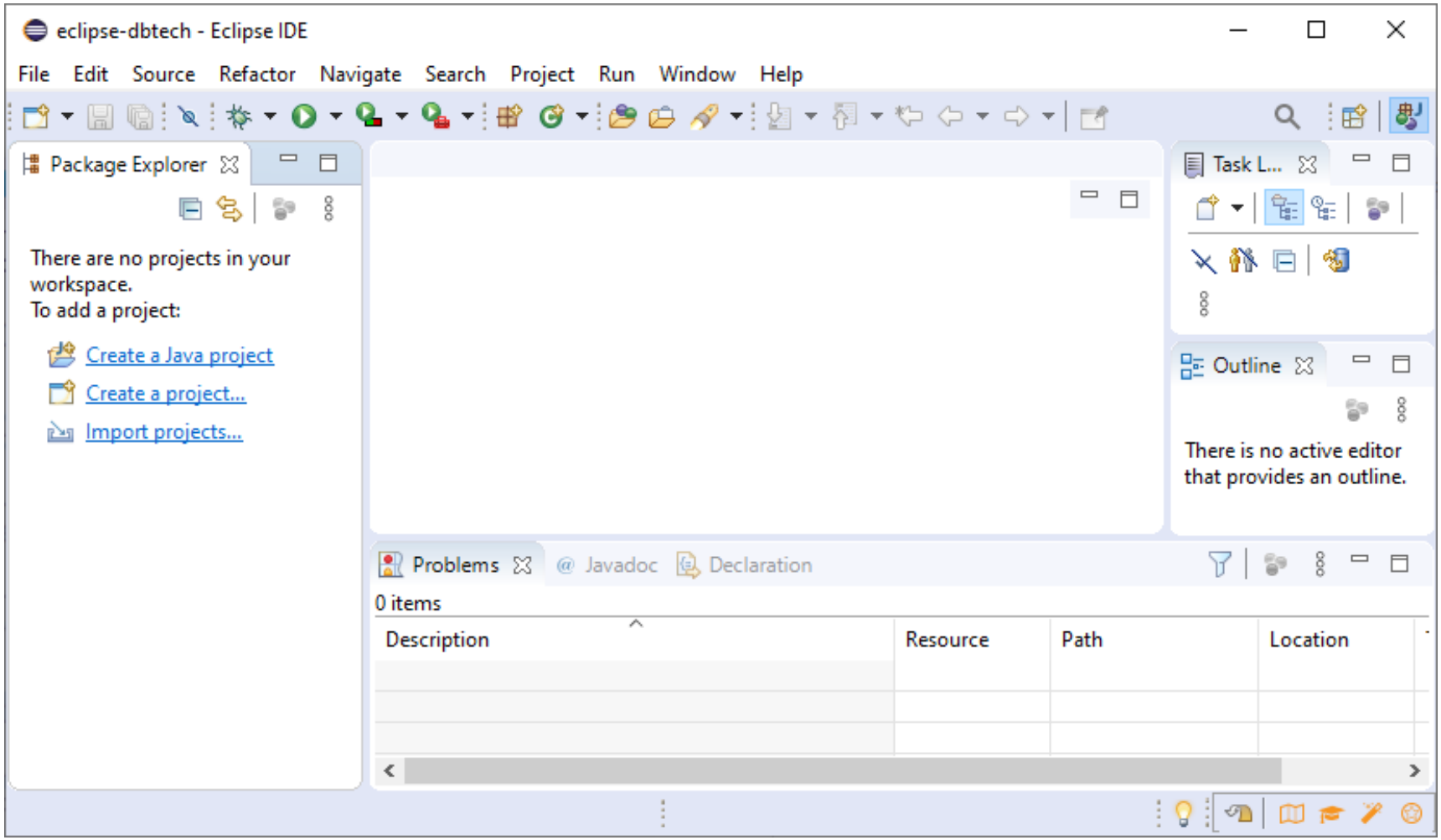


Neuer Workspace - noch nicht initialisiert



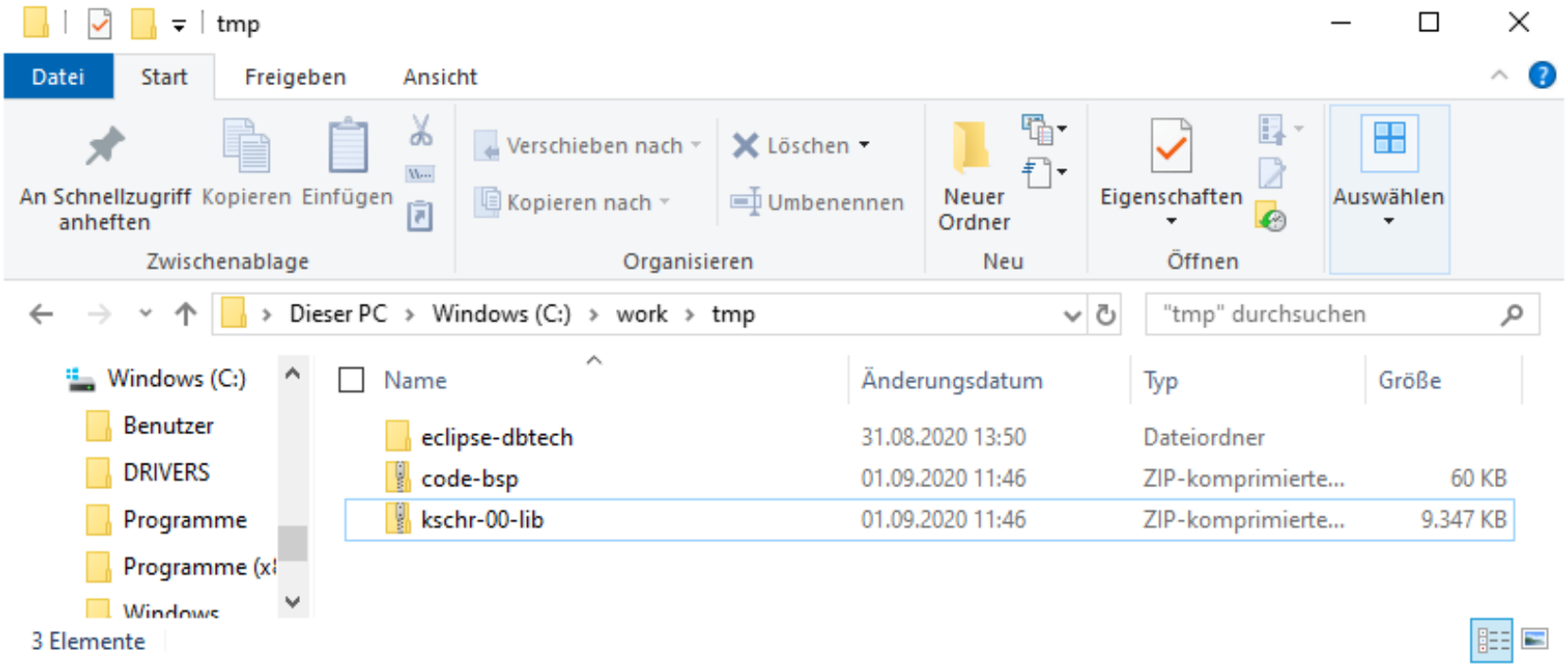
- Rechts oben auf *Workbench* klicken

Neuer Workspace - initialisiert, leer

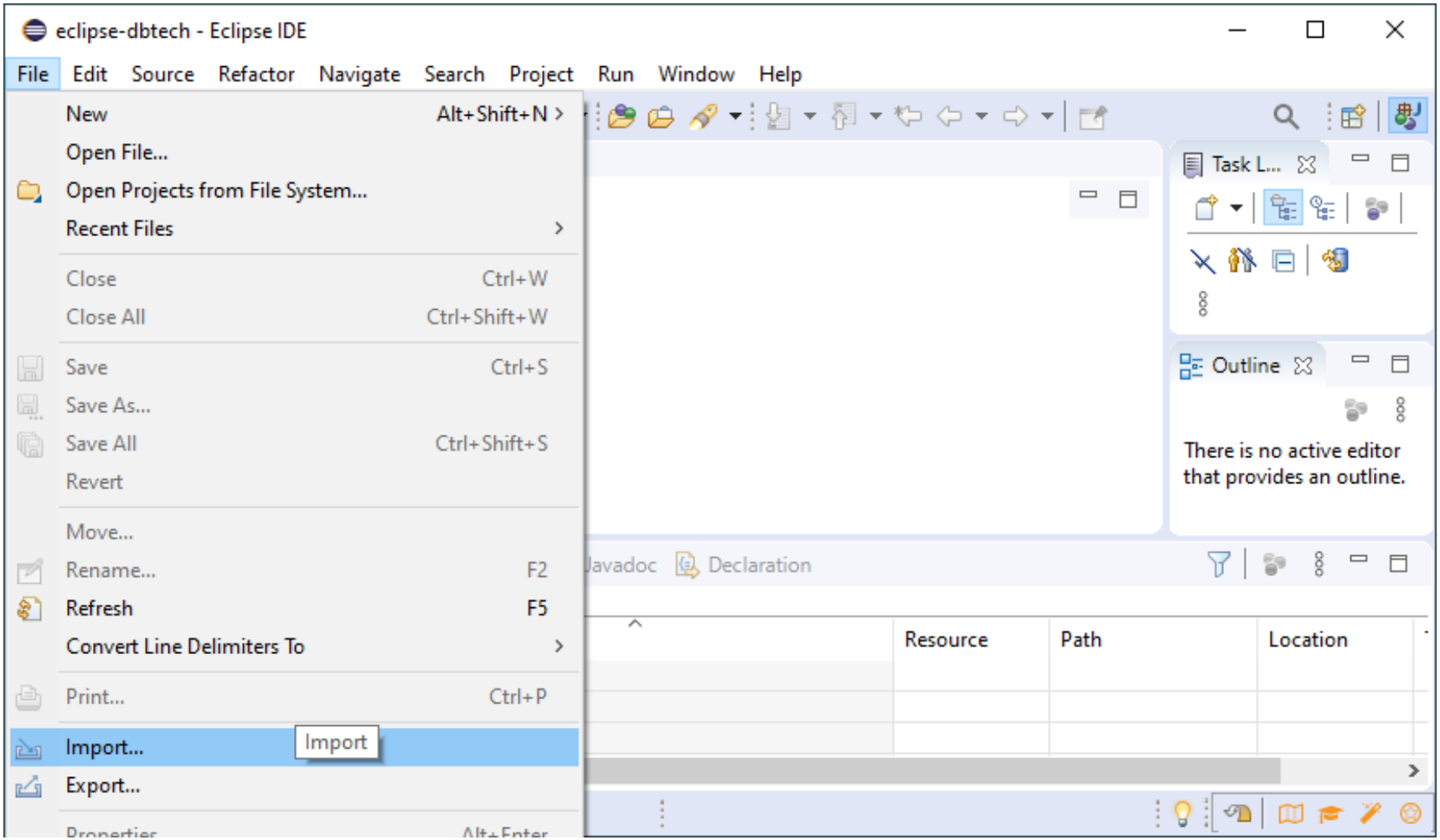


Eclipse-Projekte herunterladen

- Bibliotheksprojekt – `dbtech-lib.zip`
- Projekt mit Beispielcode – `code-bsp.zip`
- Siehe Seite *DbTech - Übung*

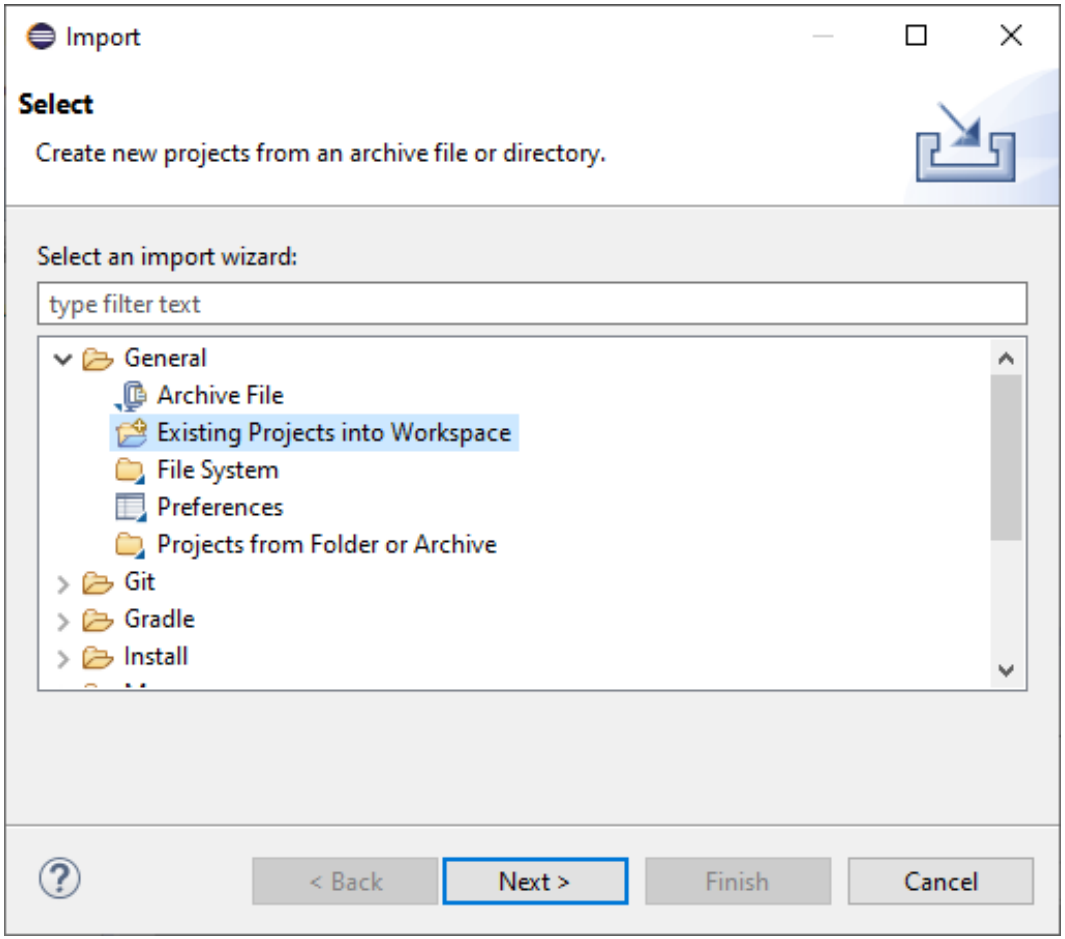


Bibliotheksprojekt importieren (1)



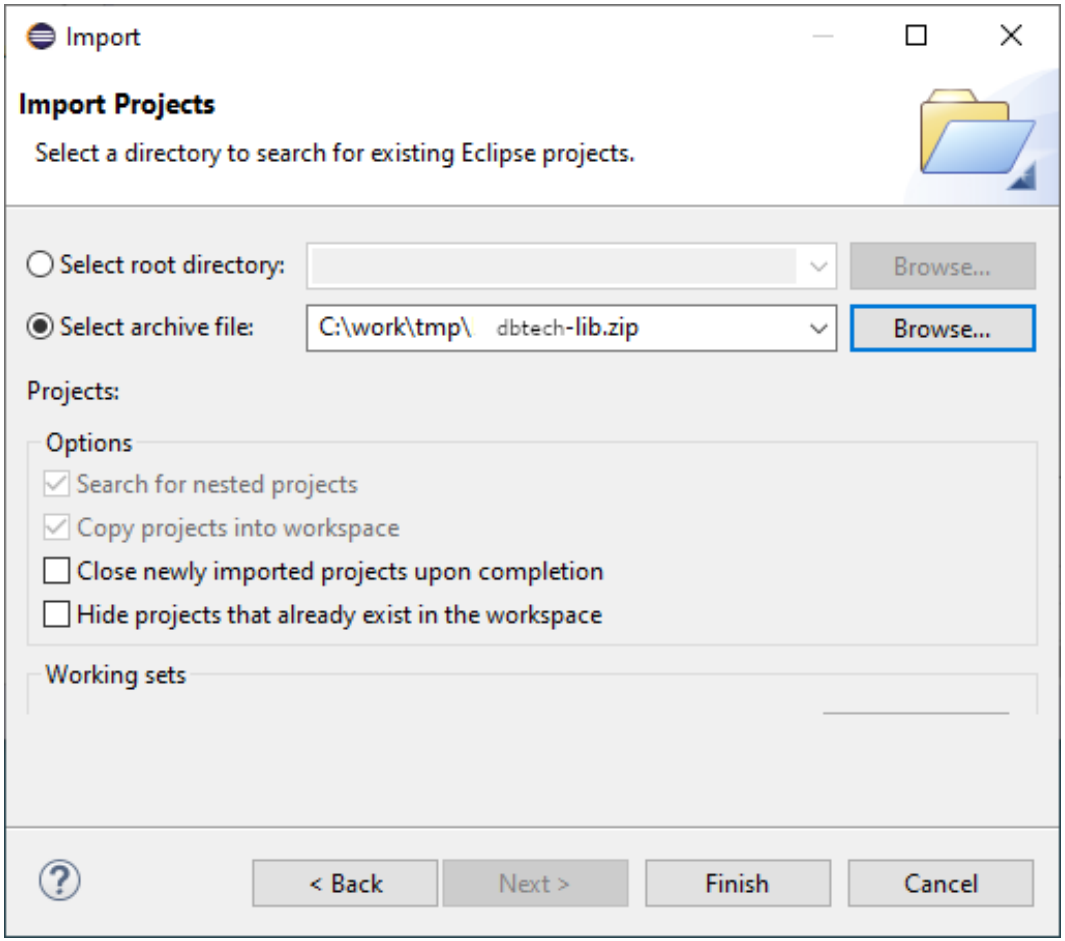
Bibliotheksprojekt importieren (2)

- *Existing Projekt into Workspace* auswählen



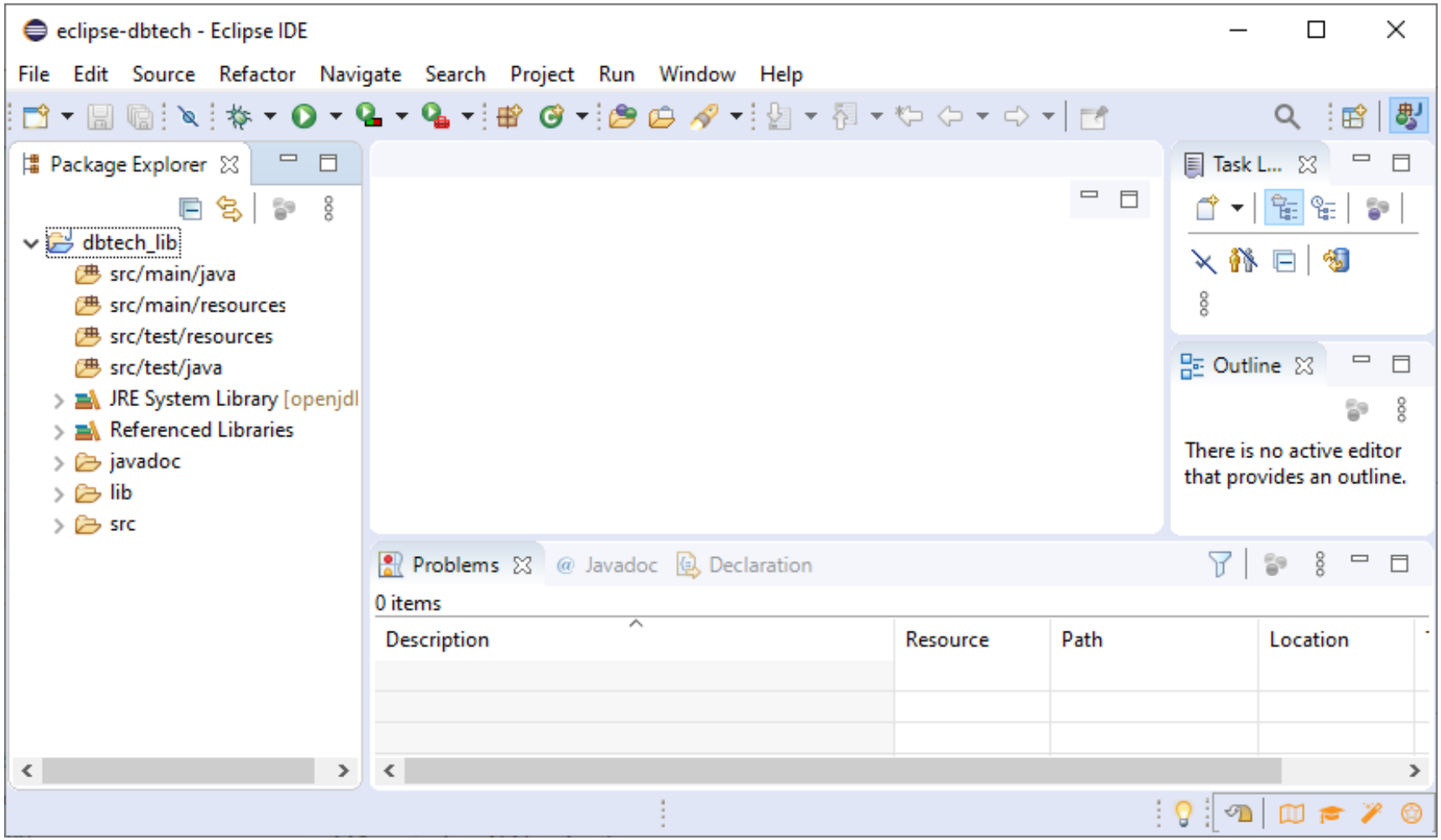
Bibliotheksprojekt importieren (3)

- *Select archive file* auswählen, dann *Browse*



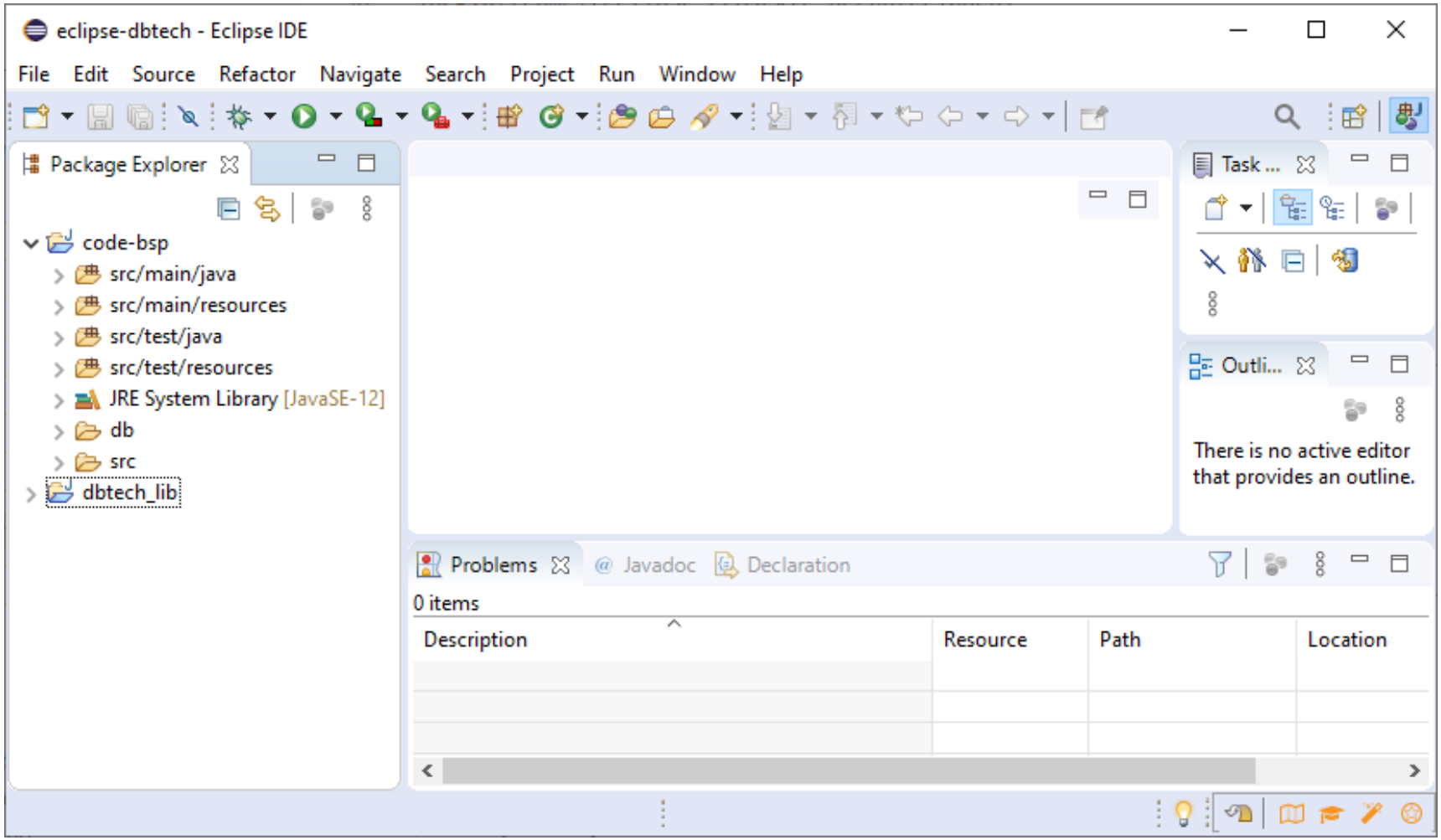
Bibliotheksprojekt importieren (3)

- *Finish* auswählen



Projekt mit Beispiel-Code importieren

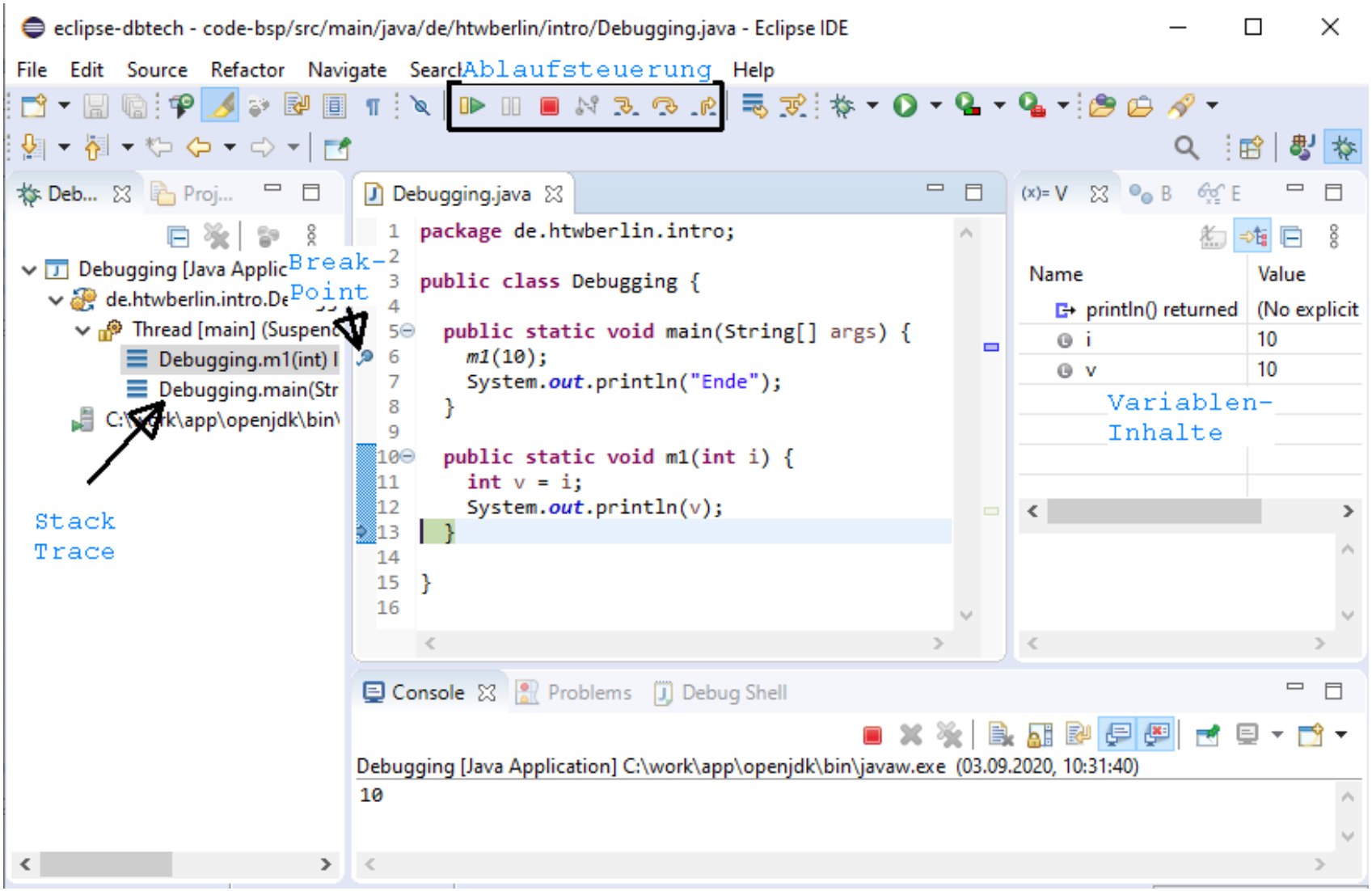
- Gleiches Vorgehen wie beim Bibliotheksprojekt



Verzeichnisstruktur eines Projekts

`src/main/java`	In diesem Unterverzeichnis befinden sich die Java-Pakete und Dateien, die den Anwendungscode enthalten. Nur dieser wird im Produkt bereitgestellt.
`src/main/resources`	In diesem Unterverzeichnis befinden sich weitere Dateien (z.B. Bilder, Konfigurationen), die in der Anwendung verwendet werden.
`src/test/java`	In diesem Unterverzeichnis befinden sich die Java-Pakete und Dateien, die den Testscode enthalten. Wird im Allgemeninen nicht mit ausgeliefert
`src/test/resources`	In diesem Unterverzeichnis befinden sich weitere Dateien (z.B. Testdaten), die in den Tests verwendet werden.

Interaktives schrittweises Ausführen des Programms



Programm mit Protokollanweisungen

```
public class Logging {
  private static final Logger L = LoggerFactory.getLogger(Logging.class);

  public static void main(String[] args) {
    L.info("Start");
    m1(10);
    L.info("Ende");
  }

  public static void m1(int i) {
    L.info("Start");
    L.debug("i: " + i);
    int v = i;
    System.out.println(v);
    L.info("Ende");
 }
}
```

Checked Exceptions vs Runtime Exceptions

```
public class RaumException extends RuntimeException {
  private static final long serialVersionUID = 1L;

  public RaumException() {
  }

  public RaumException(Throwable t) {
    super(t);
  }

  public RaumException(String msg) {
    super(msg);
  }
```

Interface

```java
package de.htwberlin.raum;
import java.sql.Connection;

public interface IRaumService {
  void setConnection(Connection connection);
  Integer findAnzahlPlaetzeInRaum(int rid);
}
```

Implementierende Klasse (1)

```
public class RaumService implements IRaumService {
  private static final Logger L = LoggerFactory.getLogger(RaumService.class);
  private Connection connection;

  @Override
  public void setConnection(Connection connection) {
    this.connection = connection;
    L.debug("connection set");
  }

  protected Connection useConnection() {
    if (connection != null) {
      return this.connection;
    } else {
      throw new RuntimeException("Connection not existing");
    }
  }

  @Override
  public Integer findAnzahlPlaetzeInRaum(int rid) {...}
}
```

Implementierende Klasse (2)

```
public class RaumService implements IRaumService {
  ...

  @Override
  public Integer findAnzahlPlaetzeInRaum(int rid) {
    String sql = "select AnzahlSitze from Raum where RID=?";
    L.info(sql);
    try (PreparedStatement ps = useConnection().prepareStatement(sql)) {
      ps.setInt(1, rid);
      try (ResultSet rs = ps.executeQuery()) {
        if (rs.next()) {
          return rs.getInt("AnzahlSitze");
        } else {
          throw new RaumException("rid doesn't exist in db: " + rid);
        }
      }
    } catch (SQLException e) {
      L.error("", e);
      throw new DataException(e);
    }
  }
}
```

Main

```
public class RaumMain {
  private static final Logger L = LoggerFactory.getLogger(RaumMain.class);

  public static void main(String[] args) {
    IRaumService rs = new RaumService();

    try (Connection connection = JdbcUtils.getConnectionViaDriverManager(
            DbCred.url, DbCred.user, DbCred.password)) {
      rs.setConnection(connection);
      Integer rid = 1;
      int anzahl = rs.findAnzahlPlaetzeInRaum(rid);
      System.out.println(anzahl);
    } catch (SQLException e) {
      L.error("Verbindungsaufbau gescheitert", e);
    } catch (DataException e) {
      L.error("DataException");
    }
  }
}
```