UECS2344 Software Design: Practical 12

Object Persistence - Persisting data in database

From https://www.objectdb.com/

ObjectDB is a software component for developing Java database applications using the Java Persistence API (JPA). It contains a database management system. It allows Java applications to store data for entity classes persistently in a database. The JPA is used in Java code to specify database manipulations operations.

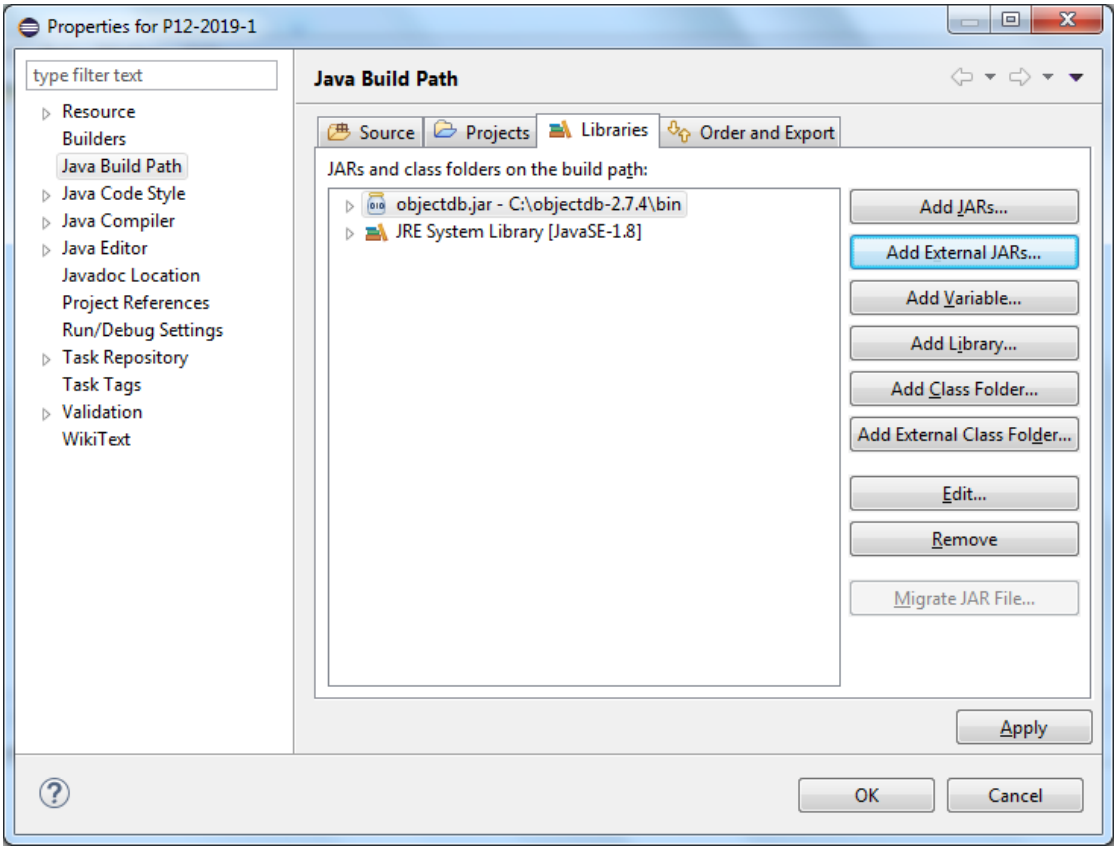
Part A

The Student Registry application in Practical 3 is revised to handle persistence of Student objects in a database. In order to use the ObjectDB component, take the following steps:

1. Download and extract the ObjectDB distribution zip file into the C: drive.

2. Create a new Java project. In the dialog window, type the name of the project, and then click Next.

3. In the second dialog window (Java Settings), click on the Libraries tab. Click on Add External JARs… and select the objectdb.jar file from the *bin* subdirectory of the ObjectDB installation directory. Then click Finish.



4. The application code is as follows.

```
// Student.java

import java.io.Serializable;
import javax.persistence.*;

@Entity
public class Student implements Serializable {

	private static final long serialVersionUID = 1L;

	@Id @GeneratedValue
	private long id;

	private int studentId;

	private String name;

	// no-args constructor
	public Student() {
	}

	public Student(String name, int studentId) {
		this.name = name;
		this.studentId = studentId;
	}

	public Long getId() {
		return id;
	}

	public int getStudentId() {
		return studentId;
	}

	public String getName() {
		return name;
	}
}
```

```
// IStudentRegisrty.java

import java.util.List;

public interface IStudentRegistry {

	public void addStudent(String name, int id);

	public Student searchStudent(String name);

	public int getNumberOfStudents();

	public List<Student> getStudents();

	public void close();
```

```
}
```

```
// StudentRegistryDB.java

import javax.persistence.*;

import java.util.*;

public class StudentRegistryDB implements IStudentRegistry {

	private EntityManagerFactory emf;

	private EntityManager em;


	public StudentRegistryDB() {
		// Create an EntityManagerFactory for database students.odb.
		emf = Persistence.createEntityManagerFactory
				("$objectdb/db/students.odb");

		// Create a new EntityManager.
		em = emf.createEntityManager();
	}

	// addStudent() method
	public void addStudent(String name, int id) {

		Student aStudent = new Student(name, id);

		// Store new Student object in the database
		em.getTransaction().begin();
		em.persist(aStudent);
		em.getTransaction().commit();
	}

	// searchStudent() method
	public Student searchStudent(String name) {

		Student theStudent;

		// Retrieve Student object from the database with given name
		TypedQuery<Student> query =
			em.createQuery("SELECT s FROM Student s WHERE s.name= '"
					+ name + "'", Student.class);

		List<Student> results = query.getResultList();

		if (results.size()!= 0)
			theStudent = results.get(0);
		else
			theStudent = null;

		return theStudent;
	}

	// getNumberOfStudents() method
	public int getNumberOfStudents() {
```

```
            // Retrieve count of Student objects in the database
            TypedQuery<Long> query = em.createQuery
                        ("SELECT COUNT(s) FROM Student s", Long.class);
            long count = (long) query.getSingleResult();

            return (int)count;
      }

      // getStudents() method
      public ArrayList<Student> getStudents() {

            // Retrieve all Student objects from the database
            TypedQuery<Student> query =
                        em.createQuery("SELECT s FROM Student s",
                              Student.class);

            List<Student> results = query.getResultList();

            ArrayList<Student> students = (ArrayList<Student>) results;

            return students;
      }

      public void close() {
            // Close the database connection
            em.close();
            emf.close();
      }
}
```

```
// RegisrtyApp.java

import java.util.List;
import java.util.Scanner;

public class RegistryApp {

    private static Scanner scanner;

    private static IStudentRegistry studentList;

    public static void main(String[] args) {

        studentList = new StudentRegistryDB();

        scanner = new Scanner(System.in);

         int choice;

        do {
            System.out.println("Do you want to:");
            System.out.println("1. Register new student");
            System.out.println("2. Search for student");
            System.out.println("3. Display all students");
            System.out.println("4. Exit");

            System.out.print("Enter your choice (1-4): ");
            choice = scanner.nextInt();
```

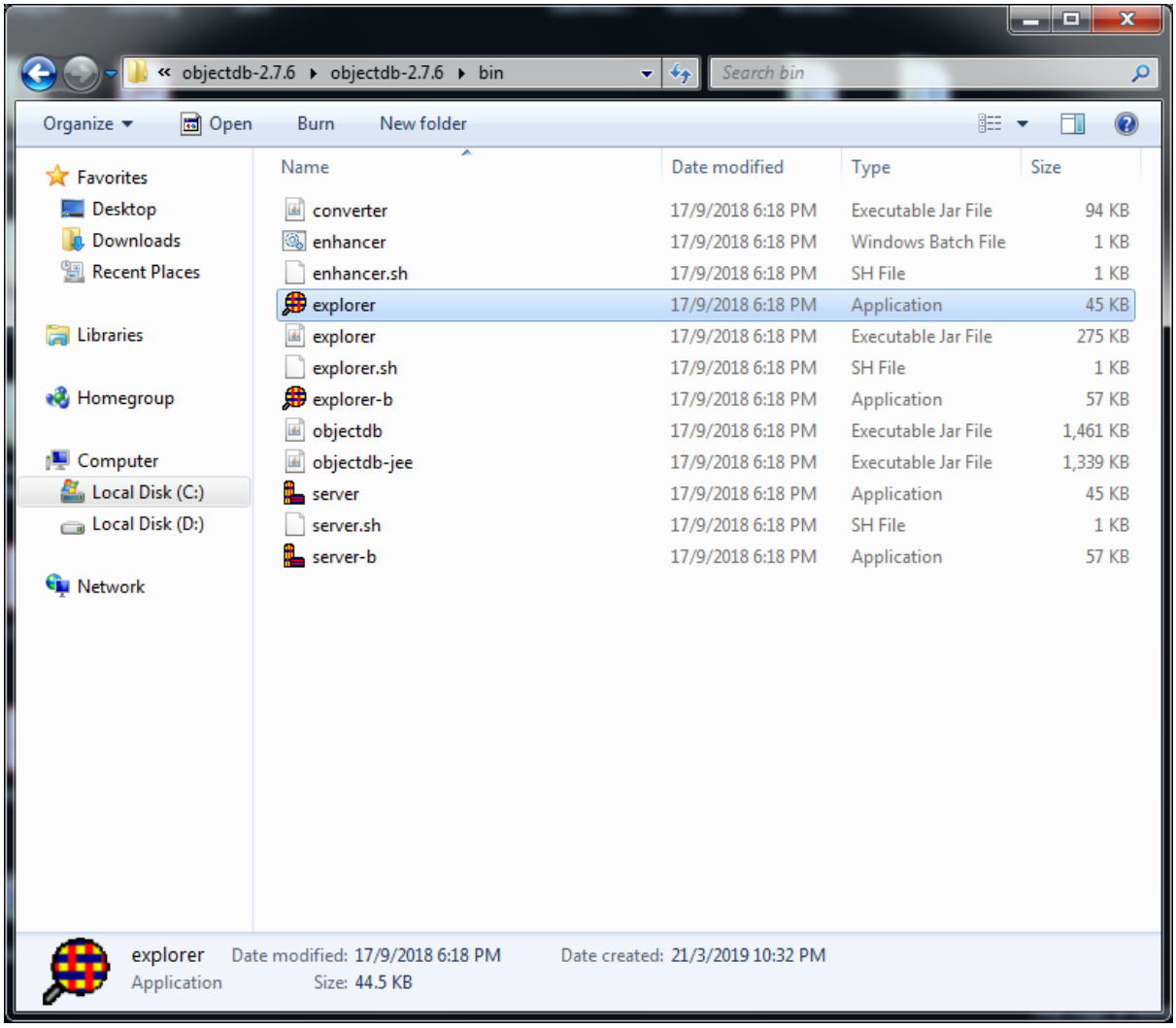
```
            // Clear ENTER key after integer input
            String skip = scanner.nextLine();

            while (choice < 1 || choice > 4)
            {
                  System.out.println("Invalid choice.");
                  System.out.print("Enter your choice (1-4): ");
                  choice = scanner.nextInt();
                  // Clear ENTER key after integer input
                  skip = scanner.nextLine();
            }

             switch(choice) {
                  case 1: addStudent(); break;
                  case 2: searchStudent(); break;
                  case 3: displayStudents(); break;
             }
             System.out.println();

        } while (choice != 4);

         studentList.close();

    }

    public static void addStudent() {
        System.out.print("Enter student name: ");
        String theName = scanner.nextLine();
        System.out.println("Enter student id: ");
        int theId = scanner.nextInt();

        studentList.addStudent(theName, theId);
        System.out.println("Student added");
        System.out.println();
    }

    public static void searchStudent() {
        System.out.print("Enter name of student: ");
        String theName = scanner.nextLine();

        Student theStudent = studentList.searchStudent(theName);
        if (theStudent == null)
            System.out.println("No student with that name found");
        else
            System.out.println("Name: " + theStudent.getName()
                        + "\tId: " + theStudent.getId());
        System.out.println();
     }

     public static void displayStudents() {
        List<Student> theStudents = studentList.getStudents();
        Student aStudent;
        for (int i=0; i< theStudents.size(); i++) {
            aStudent = theStudents.get(i);
            System.out.println("Name: " + aStudent.getName()
                        + "\tId: " + aStudent.getId());
        }
    }
}
```

Using ObjectDB Explorer to view the objects in the database

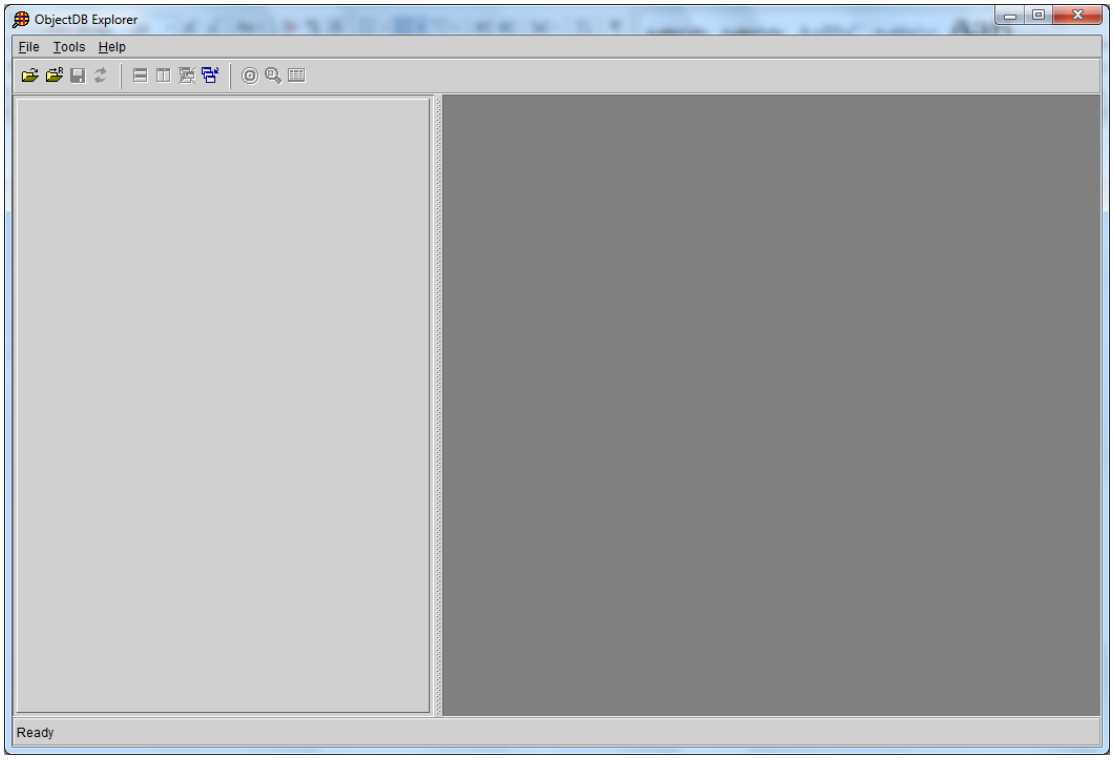
After you run the application and have created Student objects, you use the ObjectDB Explorer to view the Student objects in the database.

1. Open the objectdb directory / folder.

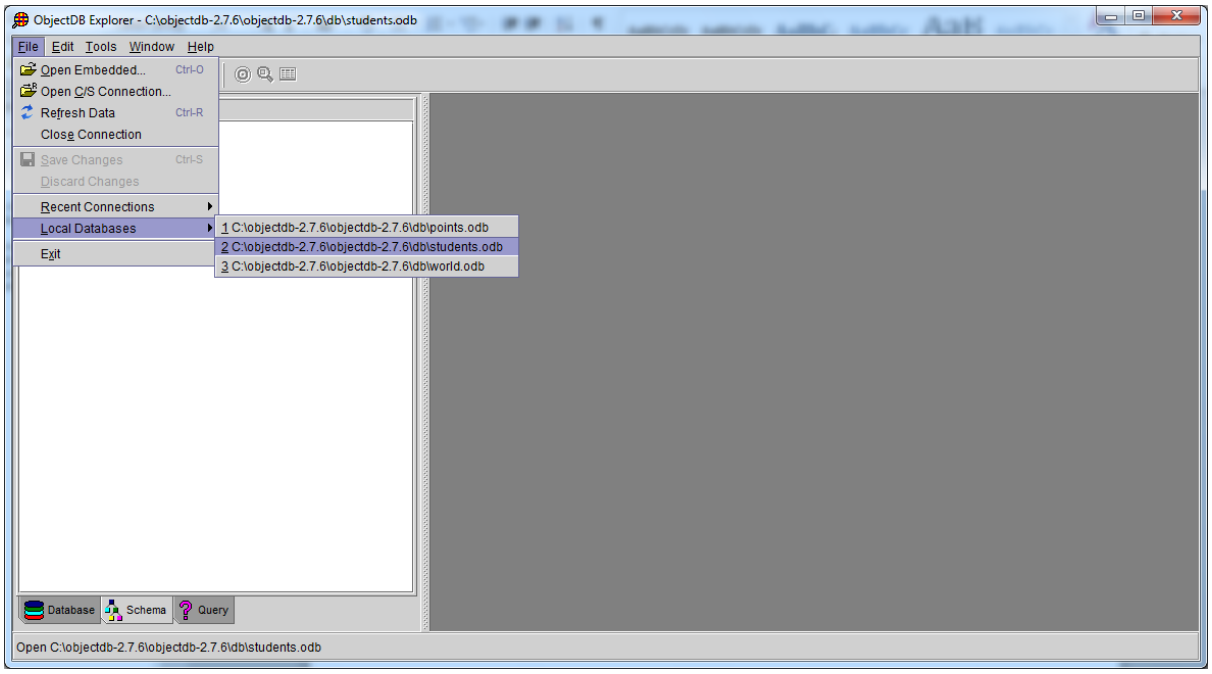
2. Open the bin subdirectory.



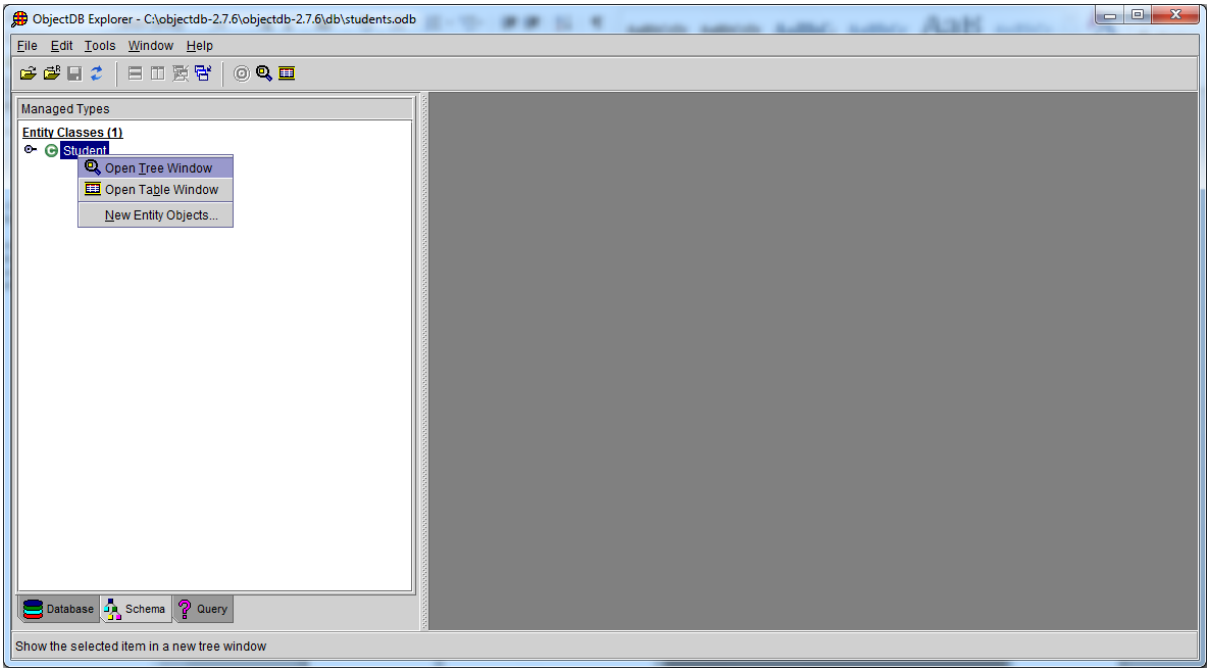
3. Double-click on explorer file. The ObjectDB Explorer application window opens.



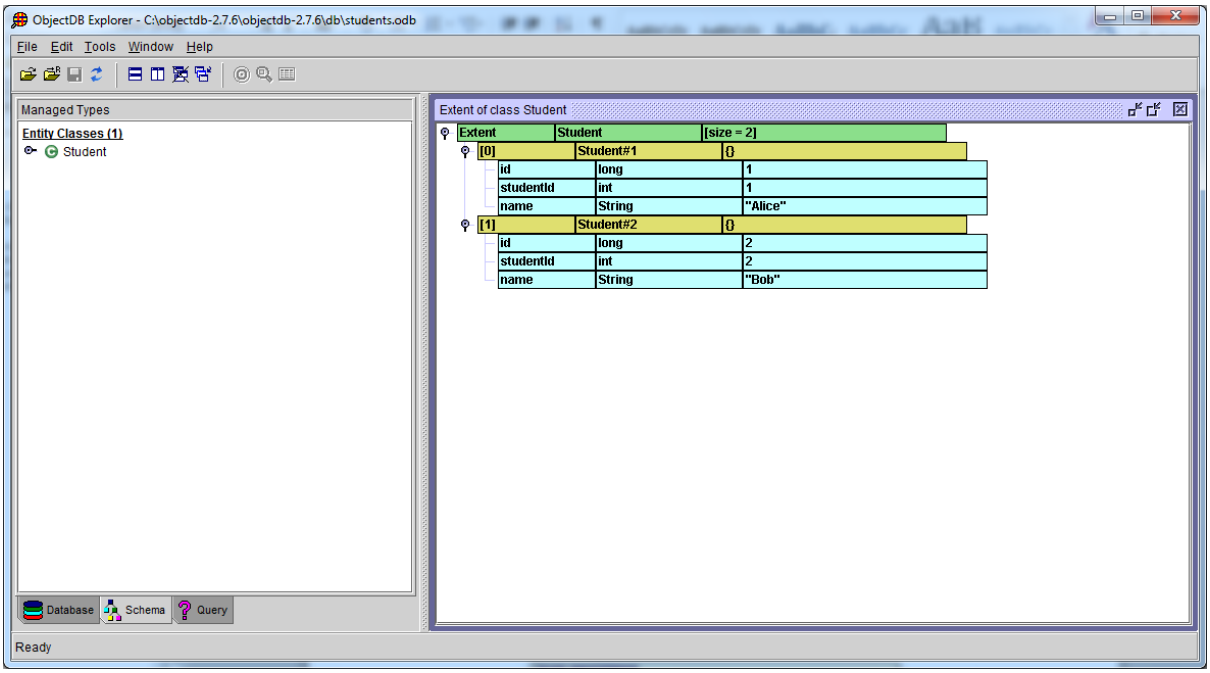
4. Select File -> Local Databases - > . . . students.odb (this is the newly created database).



5. Right-click on the Student entity class and select Open Tree Window.

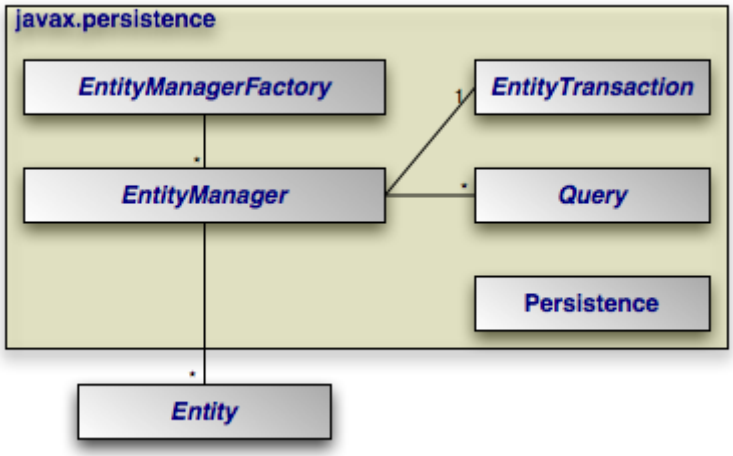


6. The Student objects that you have created are displayed.



Java Persistence API

From https://openjpa.apache.org/builds/1.2.3/apache-openjpa/docs/jpa_overview_why.html
The diagram below illustrates the relationships between the primary components of the JPA architecture.



From http://www.objectdb.com/java/jpa/persistence/store

An entity class is an ordinary user defined Java class whose instances can be stored in the database. The easy way to declare a class as an entity is to mark it with the Entity annotation and make it implement Serializable interface:

```
@Entity
public class Student implements Serializable {
. . .
}
```

EntityManager provides a connection to a database and the functionality for performing operations on the database. EntityManagerFactory is used to create EntityManager instances. Persistence class contains static helper method to obtain EntityManagerFactory instances.

The following code shows how to obtain an EntityManager instance using a EntityManagerFactory:

```
emf = Persistence.createEntityManagerFactory("$objectdb/db/students.odb");
em = emf.createEntityManager();
```

When the application is finished using the EntityManager and EntityManagerFactory it has to be closed. The following code shows how to close them:

```
em.close();
emf.close();
```

Operations that modify the content of a database require active transactions. Transactions are managed by an EntityTransaction instance obtained from EntityManager. Operations that modify the database are enclosed in transactions operations of begin and commit.

The following code stores a newly created Student object:

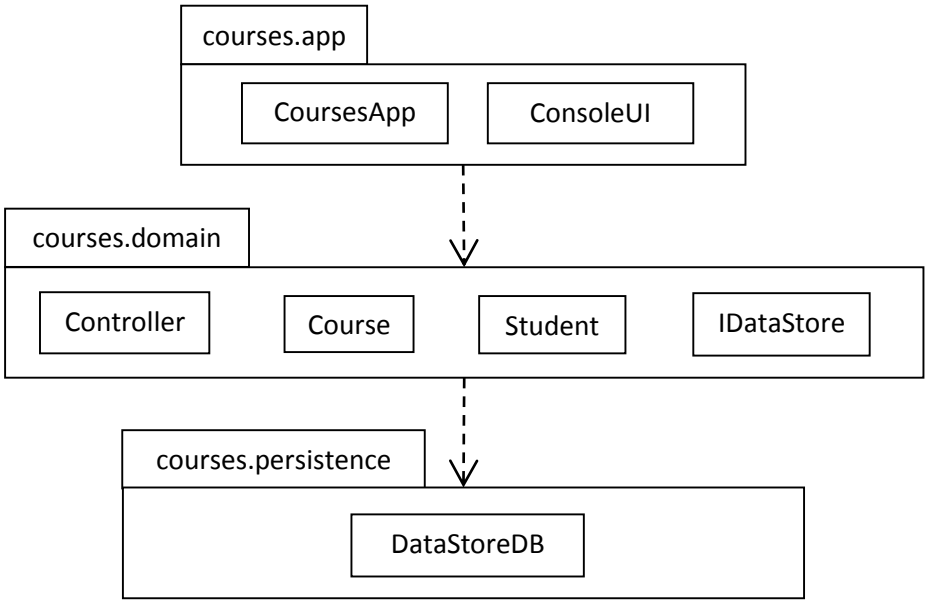
```
// Store new Student object in the database
Student aStudent = new Student(name, id);
em.getTransaction().begin();
em.persist(aStudent);
em.getTransaction().commit();
```

The following code uses a TypedQuery<> instance to retrieve a set of Student objects:

```
// Retrieve all Student objects from the database
TypedQuery<Student> query = em.createQuery("SELECT s FROM Student s",
                                           Student.class);
List<Student> results = query.getResultList();
ArrayList<Student> students = (ArrayList<Student>) results;
```

Part B

The Course Management System in Practical 6 (Iteration 1 and 2) is revised to handle persistence of the Entity objects. The classes in the system are organised into 3 layers (packages)



The system code is as follows:

```
//Student.java

package courses.domain;

import java.io.Serializable;
import javax.persistence.*;

@Entity
public class Student implements Serializable {

	private static final long serialVersionUID = 1L;
```

```
	@Id @GeneratedValue
	private long id;

	private int studentId;

	private String name;

	// no-args constructor
	public Student() {
	}

	public Student(String name, int studentId) {
		this.name = name;
		this.studentId = studentId;
	}

	public Long getId() {
		return id;
	}

	public int getStudentId() {
		return studentId;
	}

	public String getName() {
		return name;
	}
}
```

```
// Course.java

package courses.domain;

import java.io.Serializable;
import javax.persistence.*;

import java.util.List;
import java.util.ArrayList;

@Entity
public class Course implements Serializable {

    private static final long serialVersionUID = 1L;

    @Id @GeneratedValue
    private long id;

	private String title;

	private String code;

	@OneToMany(cascade=CascadeType.ALL)
	private List<Student> studentsEnrolled;

	// no-args constructor
	public Course() {
	}
```

```
	public Course(String title, String code) {
		this.title = title;
		this.code = code;
		studentsEnrolled = new ArrayList<Student>();
	}

	public Long getId() {
		return id;
	}

	public String getTitle() {
		return title;
	}

	public String getCode() {
		return code;
	}

	public List<Student> getStudentsEnrolled() {
		return studentsEnrolled;
	}

	public void enroll(Student aStudent) {
		studentsEnrolled.add(aStudent);
	}

	public int getNumberEnrolled() {
		return studentsEnrolled.size();
	}

	public boolean isStudentEnrolled(String name) {
		int size = studentsEnrolled.size();
		boolean isEnrolled = false;
		int i = 0;
		Student aStudent = null;
		while (i<size && !isEnrolled) {
			aStudent = studentsEnrolled.get(i);
			if (aStudent.getName().equals(name))
				isEnrolled = true;
			else
				i++;
		}
		return isEnrolled;
	}
}
```

```
// IDataStore.java

package courses.domain;

import java.util.List;

public interface IDataStore {

	public void addCourse(Course course);

	public void addStudentForCourse(Course course, Student student);
```

```
        public Course searchCourse(String title);

        public List<Course> getAllCourses();

        public int getNumberOfCourses();

}
```

```
// DataStoreDB.java

package courses.persistence;

import javax.persistence.*;

import courses.domain.*;

import java.util.List;
import java.util.ArrayList;

public class DataStoreDB implements IDataStore {

        private EntityManagerFactory emf;

        private EntityManager em;

        public DataStoreDB() {

          // Create a EntityManagerFactory for the database
          emf = Persistence.createEntityManagerFactory(
                                "$objectdb/db/courses.odb");

          // Create a EntityManager for the database
          em = emf.createEntityManager();

          // Introduce the Entity classes if no instance yet
          em.getMetamodel().entity(Course.class);
          em.getMetamodel().entity(Student.class);
        }

        public void addCourse(Course course) {
               // Store new Course object in the database
               em.getTransaction().begin();
               em.persist(course);
               em.getTransaction().commit();
        }

        public void addStudentForCourse(Course course, Student student) {
               // Store new Student object and update Course object in database
               em.getTransaction().begin();
               em.persist(student);
               em.persist(course);
               em.getTransaction().commit();
        }

        public Course searchCourse(String code) {
               Course theCourse = null;
```

```
		// Retrieve Course object from the database with given title
		TypedQuery<Course> query =
			em.createQuery("SELECT c FROM Course c WHERE c.code= '"
					+ code + "'", Course.class);

		List<Course> results = query.getResultList();

		if (results.size()!= 0)
			theCourse = results.get(0);
		else
			theCourse = null;

		return theCourse;
	}

	public List<Course> getAllCourses() {
		// Retrieve all Course objects from the database
		TypedQuery<Course> query =
			em.createQuery("SELECT c FROM Course c",
				Course.class);

		List<Course> results = query.getResultList();

		ArrayList<Course> courses = (ArrayList<Course>) results;
		return courses;
	}

	public int getNumberOfCourses() {
		// Retrieve count of Courses objects in the database

		TypedQuery<Long> query = em.createQuery
	    ("SELECT COUNT(c) FROM Course c", Long.class);
		long count = (long) query.getSingleResult();

		return (int)count;
	}

	public void close() {
		em.close();
		emf.close();
	}
}
```

```
// Controller.java

package courses.domain;

import java.util.List;

public class Controller {

	private IDataStore dataStore;

	public Controller(IDataStore dataStore) {
		this.dataStore = dataStore;
	}
```

```
	public void setCourseList(IDataStore dataLists) {
		dataStore = dataLists;
	}

	public void addCourse(String title, String code) {
		Course course = new Course(title, code);
		dataStore.addCourse(course);
	}

	public int getNumberOfCourses() {
		return dataStore.getNumberOfCourses();
	}

	public List<Course> getAllCourses() {
		return dataStore.getAllCourses();
	}

	public Course searchCourse(String code) {
		return dataStore.searchCourse(code);
	}

	public boolean checkStudentInCourse(Course selectedCourse,
	                                    String name) {
		return selectedCourse.isStudentEnrolled(name);
	}

	public void enrollStudent(Course selectedCourse, String name,
	                          int id) {
		Student student = new Student(name, id);
		selectedCourse.enroll(student);
		dataStore.addStudentForCourse(selectedCourse, student);
	}

}
```

```
// ConsoleUI.java

package courses.app;

import java.util.List;
import java.util.Scanner;

import courses.domain.*;

public class ConsoleUI {

	private Scanner scanner;

	private Controller controller;

	public ConsoleUI() {
		scanner = new Scanner(System.in);
		controller = null;
	}

	public void setController(Controller controller) {
		this.controller = controller;
```

```
}

public void start() {

	int choice;

	do {
		System.out.println("Do you want to:");
		System.out.println("1. Display all courses");
		System.out.println("2. Add new course");
		System.out.println("3. Check if student enrolled in course");
		System.out.println("4. Enroll student in course");
		System.out.println("5. Exit");

		System.out.print("Enter your choice (1-5): ");
		choice = scanner.nextInt();
		// Clear ENTER key after integer input
		String skip = scanner.nextLine();
		while (choice < 1 || choice > 5) {
			System.out.println("Invalid choice.");
			System.out.print("Enter your choice (1-5): ");
			choice = scanner.nextInt();
			// Clear ENTER key after integer input
			skip = scanner.nextLine();
		}

		switch(choice) {
		case 1: displayAllCourses(); break;
		case 2: addCourse(); break;
		case 3: checkStudentInCourse(); break;
		case 4: enrollStudent(); break;
		case 5: break;
		}

		System.out.println();

	} while (choice != 5);
}

public void displayAllCourses() {

	int count = controller.getNumberOfCourses();

	if (count == 0)
		System.out.println("No courses to display");
	else {
		List<Course> theCourses = controller.getAllCourses();

		Course aCourse;
		for (int i=0; i<count; i++) {
			aCourse = theCourses.get(i);
			System.out.println("Code: " + aCourse.getCode()
					+ "\tTitle: " + aCourse.getTitle());
		}
	}
}

public void addCourse() {
	System.out.print("Enter course title: ");
```

```
            String theTitle = scanner.nextLine();
            System.out.print("Enter course code: ");
            String theCode = scanner.nextLine();

            controller.addCourse(theTitle, theCode);
            System.out.println("New course created");
            System.out.println();

        }

        private Course selectCourse() {
            System.out.print("Enter course code: ");
            String theCode = scanner.nextLine();

            Course selectedCourse = controller.searchCourse(theCode);

            if (selectedCourse == null)
                System.out.println("No course with code " + theCode
                                            + " found");
            else
                System.out.println("Title: "
                                + selectedCourse.getTitle());

            return selectedCourse;
        }

        public void checkStudentInCourse() {

            Course selectedCourse = selectCourse();

            if (selectedCourse != null) {
                System.out.print("Enter student name: ");
                String name = scanner.nextLine();
                if (controller.checkStudentInCourse(selectedCourse,
                                                    name))
                   System.out.println(name + " is enrolled in course");
                else
                   System.out.println(name + " is not enrolled in course");
            }
        }

        public void enrollStudent() {

            Course selectedCourse = selectCourse();

            if (selectedCourse != null) {
                System.out.print("Enter student name: ");
                String name = scanner.nextLine();
                System.out.print("Enter student id: ");
                int id = scanner.nextInt();

                controller.enrollStudent(selectedCourse, name, id);
                System.out.println("Student enrolled in course");
            }

            System.out.println();
        }
}
```

```java
// CoursesApp.java

package courses.app;

import courses.domain.*;

import courses.persistence.*;

public class CoursesApp {

	public static void main(String[] args) {

		IDataStore dataStore = new DataStoreDB();

		Controller controller = new Controller(dataStore);

		ConsoleUI userInterface = new ConsoleUI();

		userInterface.setController(controller);

		userInterface.start();

		DataStoreDB dataStoreDB = (DataStoreDB) dataStore;
		dataStoreDB.close();
    }

}
```