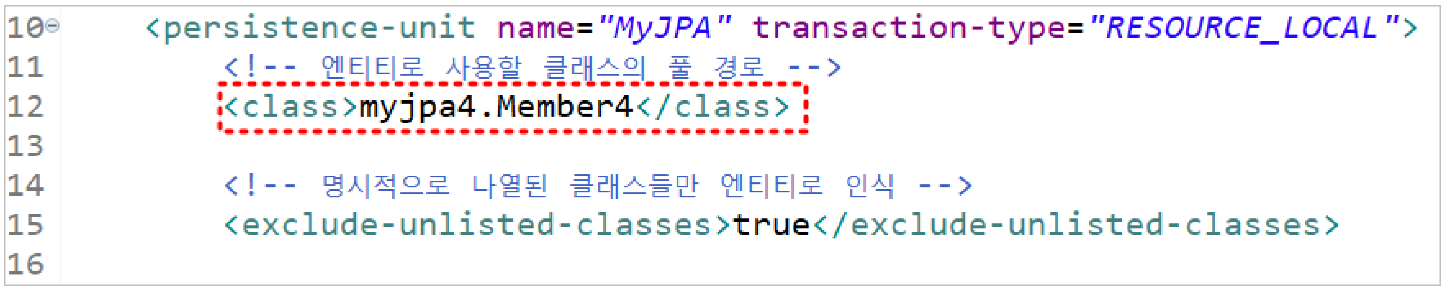
JPA 기초

두번째실습

설정파일 : resources/META-INF/persistence.xml

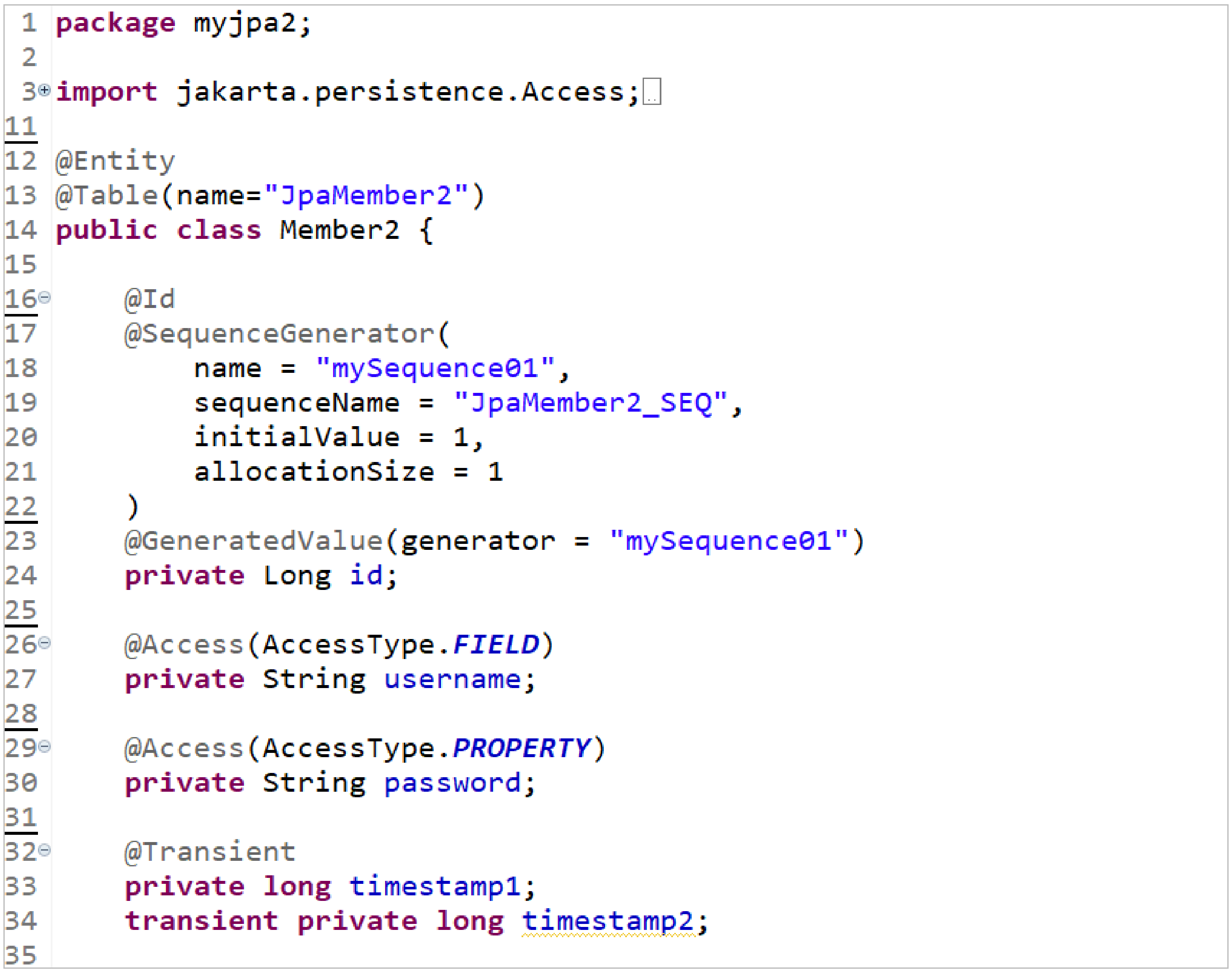
<class>myjpa4.Member4</class> ⇒ 이 부분을 myjpa2.Member2 로 수정 후 테스트

create 는 그대로 유지한다.



```xml
    <persistence-unit name="MyJPA" transaction-type="RESOURCE_LOCAL">
        <!-- 엔티티로 사용할 클래스의 풀 경로 -->
        <class>myjpa4.Member4</class>

        <!-- 명시적으로 나열된 클래스들만 엔티티로 인식 -->
        <exclude-unlisted-classes>true</exclude-unlisted-classes>

```

/B21aJPA_Basic/src/main/java/myjpa2/Member2.java



```java
package myjpa2;

import jakarta.persistence.Access;

@Entity
@Table(name="JpaMember2")
public class Member2 {

    @Id
    @SequenceGenerator(
        name = "mySequence01",
        sequenceName = "JpaMember2_SEQ",
        initialValue = 1,
        allocationSize = 1
    )
    @GeneratedValue(generator = "mySequence01")
    private Long id;

    @Access(AccessType.FIELD)
    private String username;

    @Access(AccessType.PROPERTY)
    private String password;

    @Transient
    private long timestamp1;
    transient private long timestamp2;

    public Member2() {}
    public Member2(String username, String password) {
        super();
        this.username = username;
        this.password = password;
    }

    public String getPassword() {
        return password;
    }
    public void setPassword(String password) {
        this.password = password;
    }
}

```

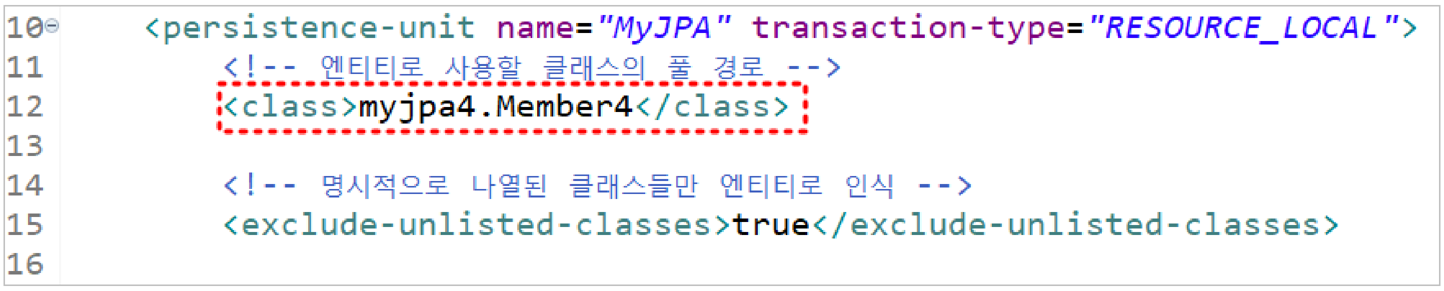
/B21aJPA_Basic/src/main/java/myjpa2/UseMember2.java

```java
package myjpa2;

import jakarta.persistence.EntityManager;

public class UseMember2 {

    public static void main(String[] args) {

        EntityManagerFactory emf =
                Persistence.createEntityManagerFactory("MyJPA");
        EntityManager em = emf.createEntityManager();
        EntityTransaction transaction = em.getTransaction();

        try {
            transaction.begin();
            Member2 member2 = new Member2("홍길동2", "1234");
            em.persist(member2);
            transaction.commit();
        }
        catch (Exception e) {
            e.printStackTrace();
            transaction.rollback();
        }
        finally {
            em.close();
        }

        emf.close();
    }
}
```

세번째실습

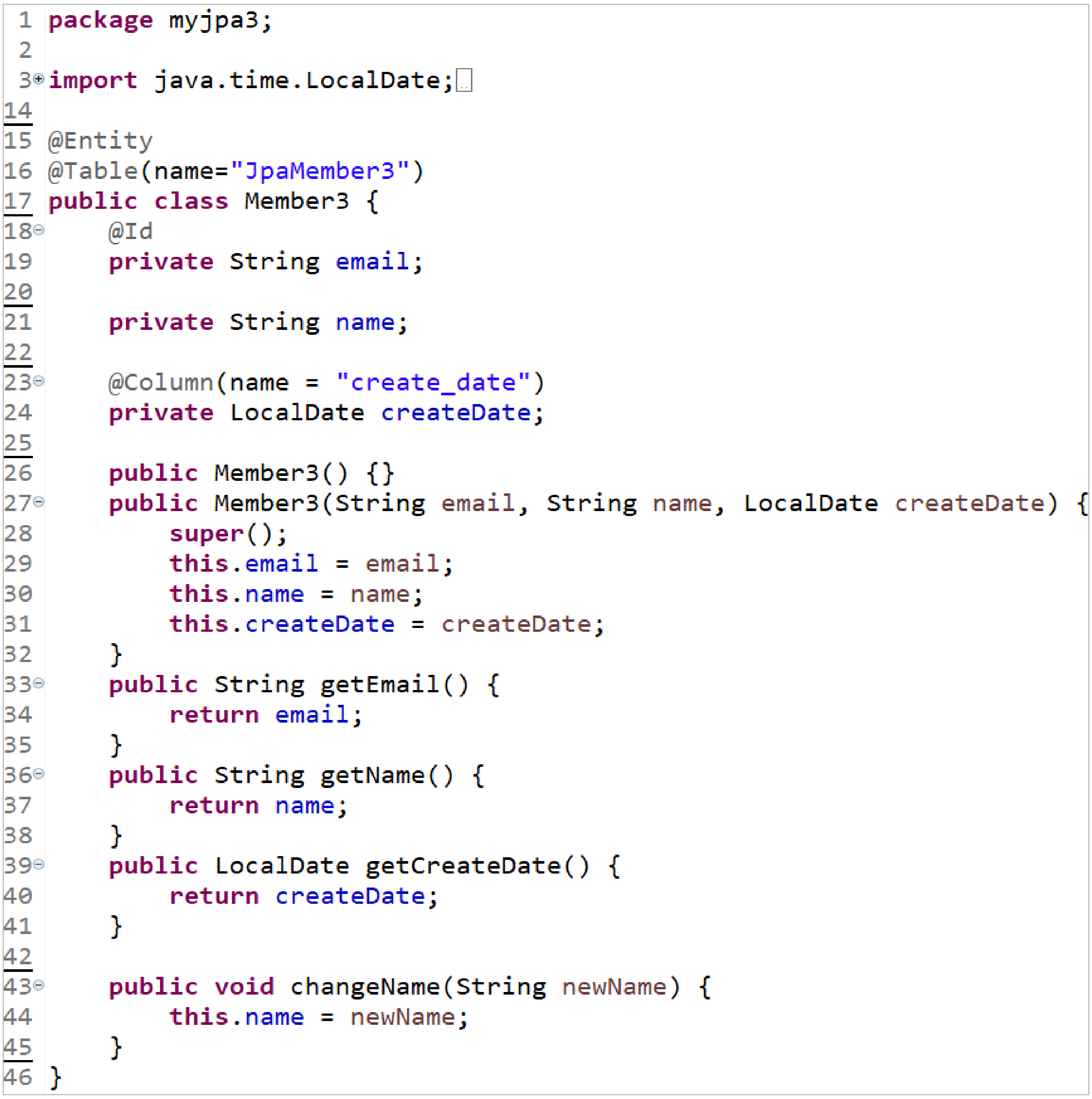
설정파일 : resources/META-INF/persistence.xml

<class>myjpa4.Member4</class> ⇒ 이 부분을 myjpa3.Member3 로 수정 후 테스트.



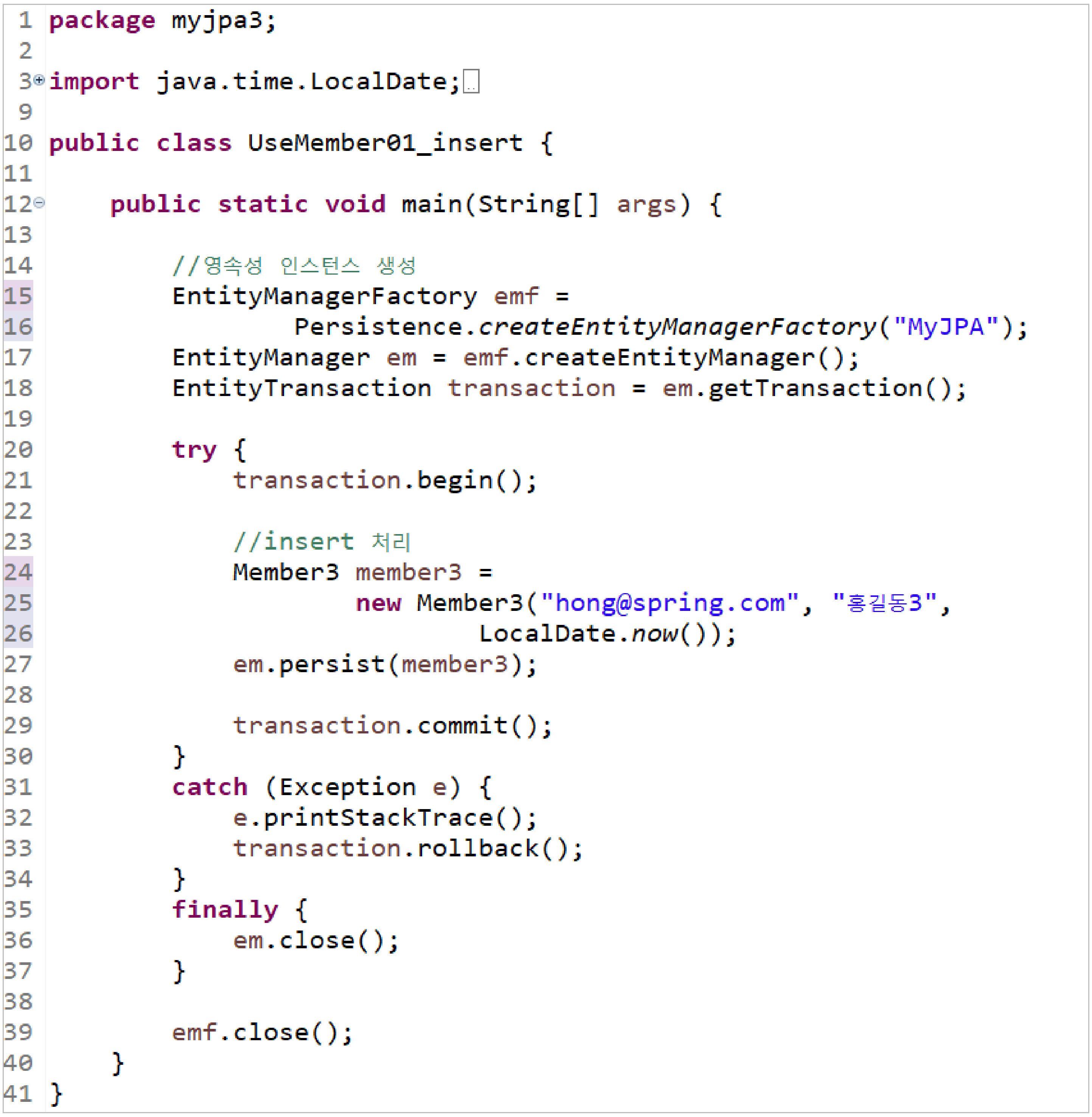
```xml
<persistence-unit name="MyJPA" transaction-type="RESOURCE_LOCAL">
    <!-- 엔티티로 사용할 클래스의 풀 경로 -->
    <class>myjpa4.Member4</class>

    <!-- 명시적으로 나열된 클래스들만 엔티티로 인식 -->
    <exclude-unlisted-classes>true</exclude-unlisted-classes>

```

/B21aJPA_Basic/src/main/java/myjpa3/Member3.java



```java
package myjpa3;

import java.time.LocalDate;

@Entity
@Table(name="JpaMember3")
public class Member3 {
    @Id
    private String email;

    private String name;

    @Column(name = "create_date")
    private LocalDate createDate;

    public Member3() {}
    public Member3(String email, String name, LocalDate createDate) {
        super();
        this.email = email;
        this.name = name;
        this.createDate = createDate;
    }
    public String getEmail() {
        return email;
    }
    public String getName() {
        return name;
    }
    public LocalDate getCreateDate() {
        return createDate;
    }

    public void changeName(String newName) {
        this.name = newName;
    }
}
```

/B21aJPA_Basic/src/main/java/myjpa3/UseMember01_insert.java

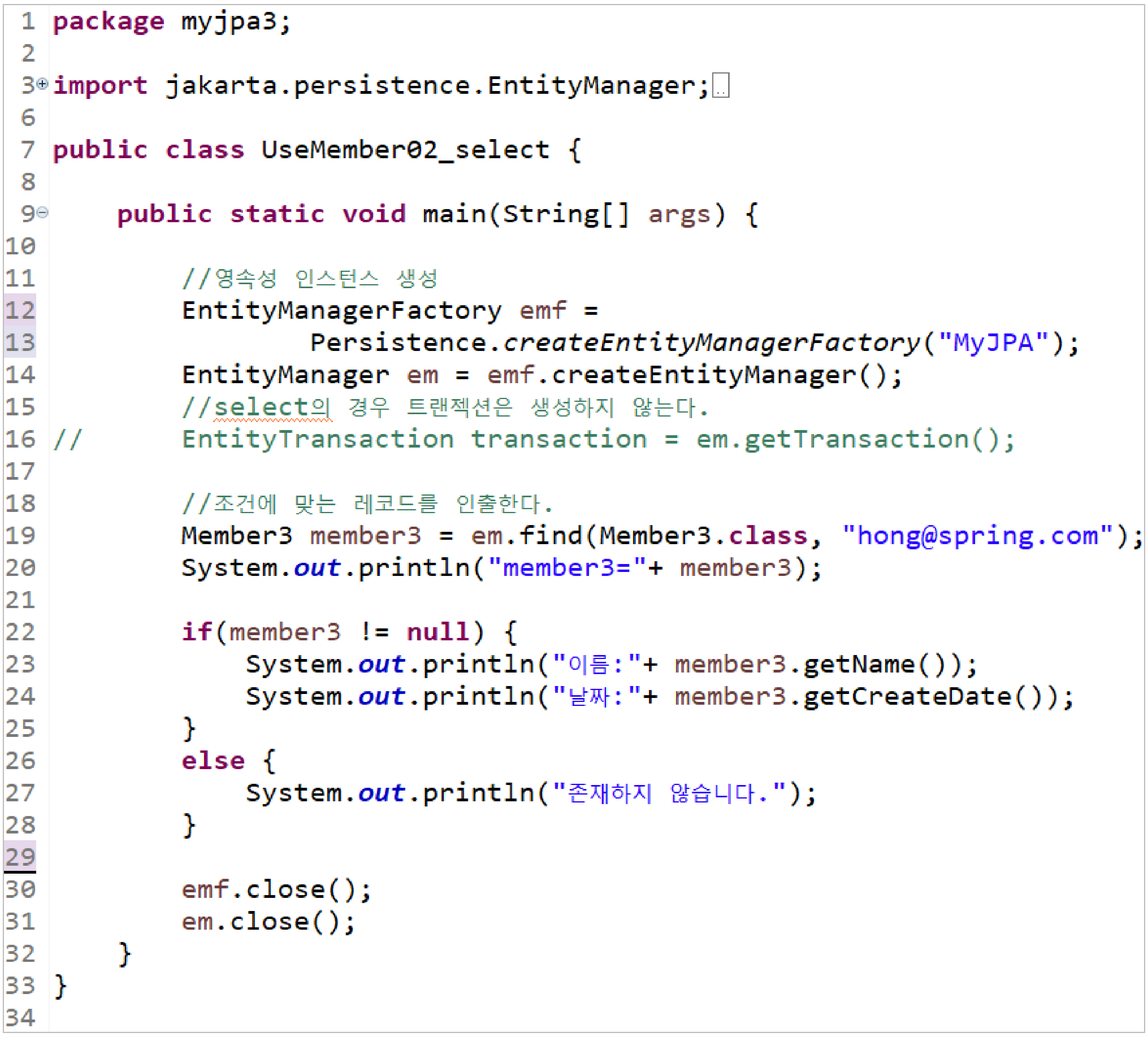


```java
package myjpa3;

import java.time.LocalDate;

public class UseMember01_insert {

    public static void main(String[] args) {

        //영속성 인스턴스 생성
        EntityManagerFactory emf =
                Persistence.createEntityManagerFactory("MyJPA");
        EntityManager em = emf.createEntityManager();
        EntityTransaction transaction = em.getTransaction();

        try {
            transaction.begin();

            //insert 처리
            Member3 member3 =
                    new Member3("hong@spring.com", "홍길동3",
                            LocalDate.now());
            em.persist(member3);

            transaction.commit();
        }
        catch (Exception e) {
            e.printStackTrace();
            transaction.rollback();
        }
        finally {
            em.close();
        }

        emf.close();
    }
}
```

설정파일 : resources/META-INF/persistence.xml

insert 이므로 create 를 유지한다.

/B21aJPA_Basic/src/main/java/myjpa3/UseMember02_select.java



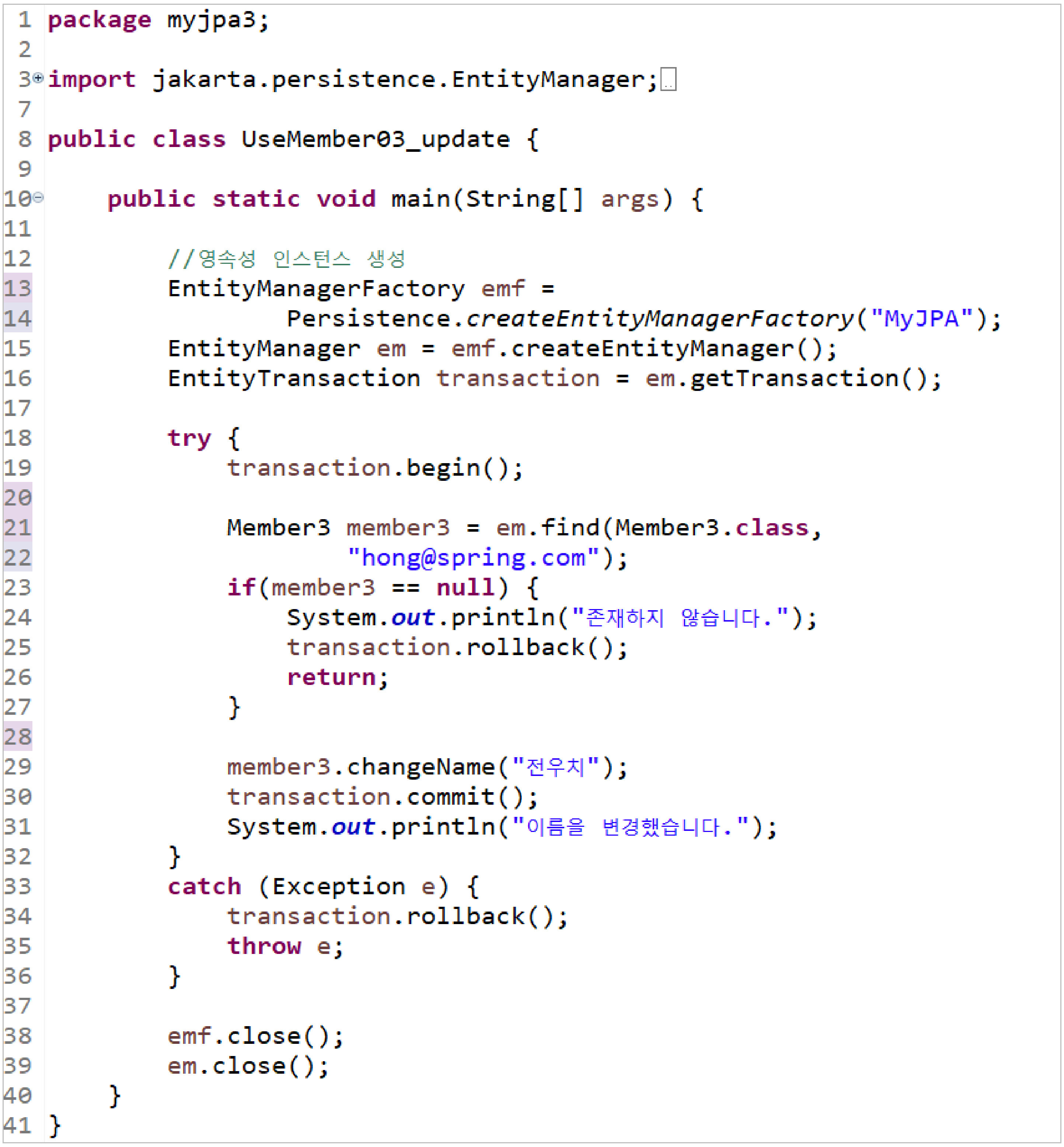
```java
package myjpa3;

import jakarta.persistence.EntityManager;

public class UseMember02_select {

    public static void main(String[] args) {

        //영속성 인스턴스 생성
        EntityManagerFactory emf =
                Persistence.createEntityManagerFactory("MyJPA");
        EntityManager em = emf.createEntityManager();
        //select의 경우 트랜젝션은 생성하지 않는다.
//      EntityTransaction transaction = em.getTransaction();

        //조건에 맞는 레코드를 인출한다.
        Member3 member3 = em.find(Member3.class, "hong@spring.com");
        System.out.println("member3="+ member3);

        if(member3 != null) {
            System.out.println("이름:"+ member3.getName());
            System.out.println("날짜:"+ member3.getCreateDate());
        }
        else {
            System.out.println("존재하지 않습니다.");
        }

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

설정파일 : resources/META-INF/persistence.xml

select 이므로 none 으로 수정 후 테스트한다.

create 를 유지하면 테이블 삭제 후 새롭게 만들어 지므로 레코드가 없다고 출력된다.

/B21aJPA_Basic/src/main/java/myjpa3/UseMember03_update.java



```java
package myjpa3;

import jakarta.persistence.EntityManager;

public class UseMember03_update {

    public static void main(String[] args) {

        //영속성 인스턴스 생성
        EntityManagerFactory emf =
                Persistence.createEntityManagerFactory("MyJPA");
        EntityManager em = emf.createEntityManager();
        EntityTransaction transaction = em.getTransaction();

        try {
            transaction.begin();

            Member3 member3 = em.find(Member3.class,
                    "hong@spring.com");
            if(member3 == null) {
                System.out.println("존재하지 않습니다.");
                transaction.rollback();
                return;
            }

            member3.changeName("전우치");
            transaction.commit();
            System.out.println("이름을 변경했습니다.");
        }
        catch (Exception e) {
            transaction.rollback();
            throw e;
        }

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

설정파일 : resources/META-INF/persistence.xml

none 으로 유지한 후 테스트한다.

/B21aJPA_Basic/src/main/java/myjpa3/UseMember04_delete.java

```java
package myjpa3;

import jakarta.persistence.EntityManager;

public class UseMember04_delete {

    public static void main(String[] args) {
        //영속성 인스턴스 생성
        EntityManagerFactory emf =
                Persistence.createEntityManagerFactory("MyJPA");
        EntityManager em = emf.createEntityManager();
        EntityTransaction transaction = em.getTransaction();

        try {
            transaction.begin();

            //조건에 맞는 레코드 검색
            Member3 member3 = em.find(Member3.class,
                    "hong@spring.com");
            if(member3 == null) {
                System.out.println("존재하지 않습니다.");
                transaction.rollback();
                return;
            }
            //레코드 삭제 및 동기화
            em.remove(member3);
            transaction.commit();
            System.out.println("삭제했습니다.");
        }
        catch (Exception e) {
            transaction.rollback();
            throw e;
        }

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

설정파일 : resources/META-INF/persistence.xml

none 으로 유지한 후 테스트한다.

네번째실습

설정파일 : resources/META-INF/persistence.xml

<class>myjpa4.Member4</class> ⇒ 이 부분을 myjpa4.Member4 로 수정. create 로 수정 후 테스트.

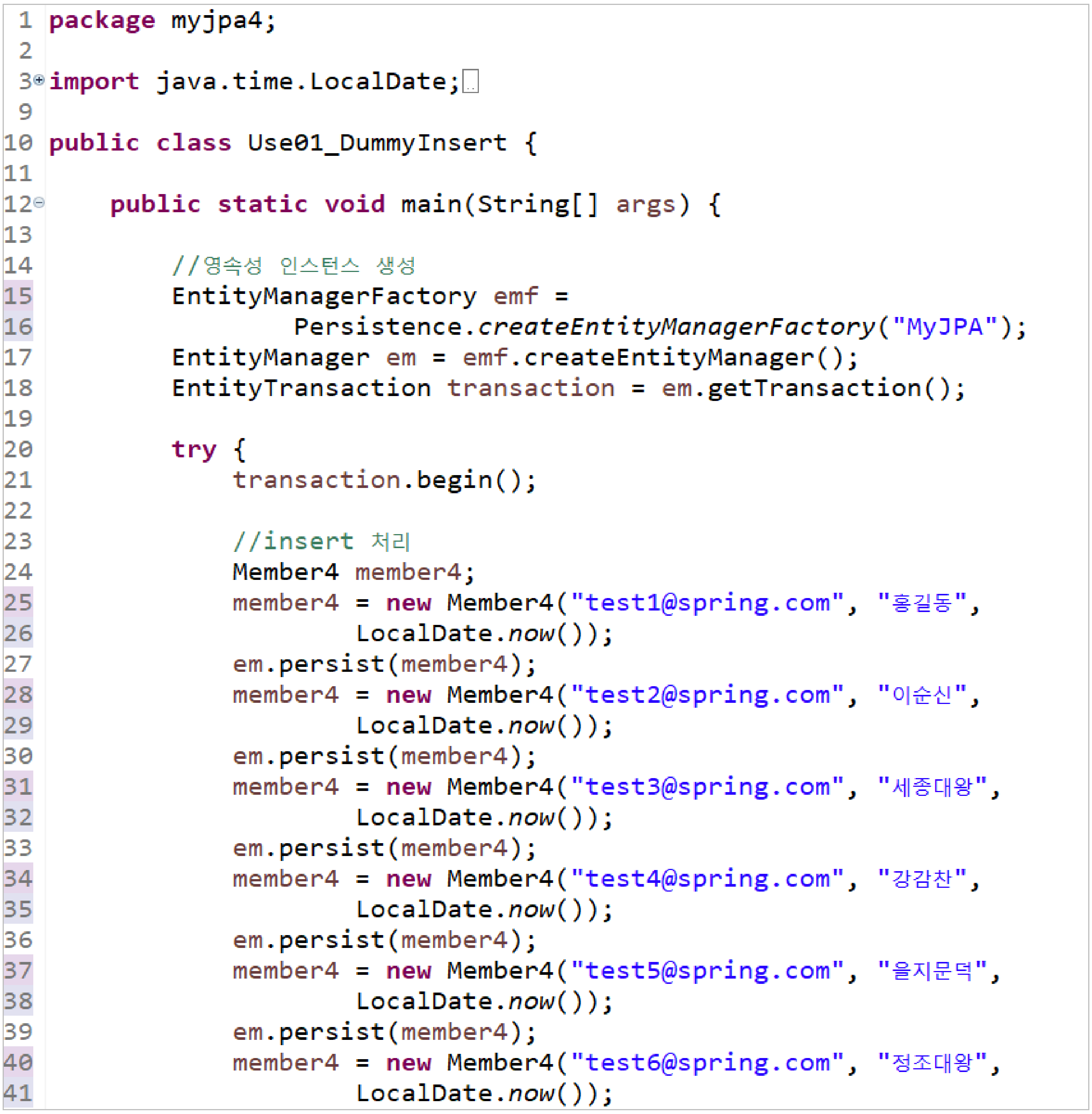
```xml
	<persistence-unit name="MyJPA" transaction-type="RESOURCE_LOCAL">
		<!-- 엔티티로 사용할 클래스의 풀 경로 -->
		<class>myjpa4.Member4</class>

		<!-- 명시적으로 나열된 클래스들만 엔티티로 인식 -->
		<exclude-unlisted-classes>true</exclude-unlisted-classes>

```

/B21aJPA_Basic/src/main/java/myjpa4/Member4.java

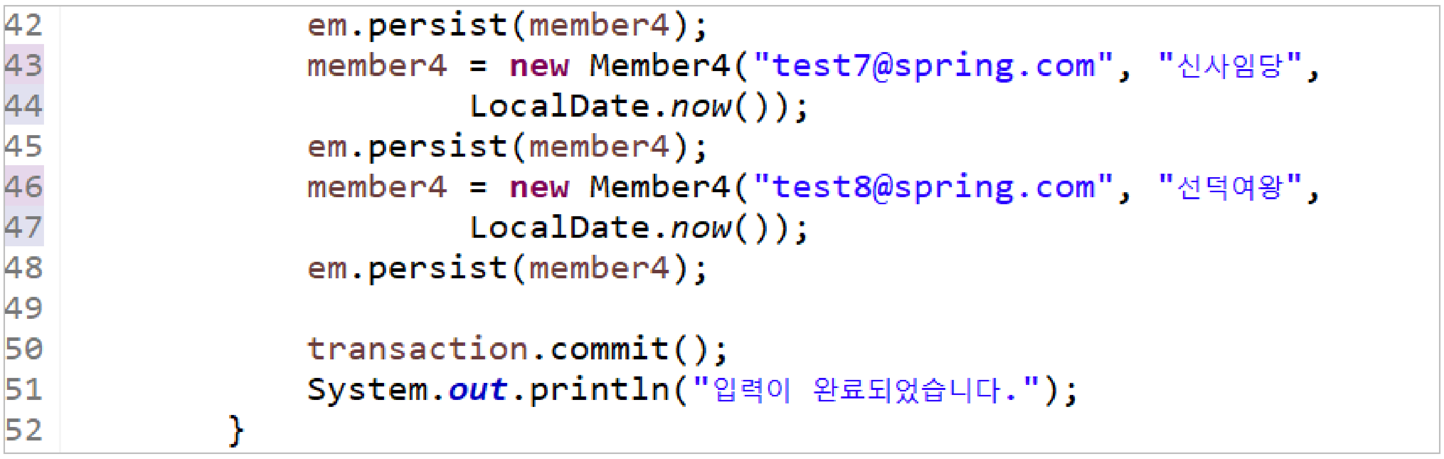
```java
package myjpa4;

import java.time.LocalDate;

@Entity
@Table(name="JpaMember4")
public class Member4 {

	@Id
	private String email;
	private String name;
	@Column(name = "create_date")
	private LocalDate createDate;

	public Member4() {}
	public Member4(String email, String name, LocalDate createDate) {
		super();
		this.email = email;
		this.name = name;
		this.createDate = createDate;
	}
	public String getEmail() {
		return email;
	}
	public String getName() {
		return name;
	}
	public LocalDate getCreateDate() {
		return createDate;
	}

	public void changeName(String newName) {
		this.name = newName;
	}
}
```

/B21aJPA_Basic/src/main/java/myjpa4/Use01_DummyInsert.java

create 인지 확인해야 함.



```java
package myjpa4;

import java.time.LocalDate;

public class Use01_DummyInsert {

    public static void main(String[] args) {

        //영속성 인스턴스 생성
        EntityManagerFactory emf =
                Persistence.createEntityManagerFactory("MyJPA");
        EntityManager em = emf.createEntityManager();
        EntityTransaction transaction = em.getTransaction();

        try {
            transaction.begin();

            //insert 처리
            Member4 member4;
            member4 = new Member4("test1@spring.com", "홍길동",
                    LocalDate.now());
            em.persist(member4);
            member4 = new Member4("test2@spring.com", "이순신",
                    LocalDate.now());
            em.persist(member4);
            member4 = new Member4("test3@spring.com", "세종대왕",
                    LocalDate.now());
            em.persist(member4);
            member4 = new Member4("test4@spring.com", "강감찬",
                    LocalDate.now());
            em.persist(member4);
            member4 = new Member4("test5@spring.com", "을지문덕",
                    LocalDate.now());
            em.persist(member4);
            member4 = new Member4("test6@spring.com", "정조대왕",
                    LocalDate.now());
```



```java
            em.persist(member4);
            member4 = new Member4("test7@spring.com", "신사임당",
                    LocalDate.now());
            em.persist(member4);
            member4 = new Member4("test8@spring.com", "선덕여왕",
                    LocalDate.now());
            em.persist(member4);

            transaction.commit();
            System.out.println("입력이 완료되었습니다.");
        }
        catch (Exception e) {
            e.printStackTrace();
            transaction.rollback();
        }
        finally {
            em.close();
        }

        //실행전 xml설정파일에서 create로 변경
        emf.close();
    }
}
```

/B21aJPA_Basic/src/main/java/myjpa4/Use02_TypedQuery.java

none 으로 수정 후 테스트

```java
package myjpa4;

import java.util.List;

public class Use02_TypedQuery {

    public static void main(String[] args) {

        EntityManagerFactory emf =
                Persistence.createEntityManagerFactory("MyJPA");
        EntityManager em = emf.createEntityManager();
        EntityTransaction transaction = em.getTransaction();

        try {
            transaction.begin();

            String SQL = "SELECT m FROM Member4 m ORDER BY m.name";
            TypedQuery<Member4> query =
                    em.createQuery(SQL, Member4.class);
            List<Member4> result = query.getResultList();

            transaction.commit();

            if(result.isEmpty()) {
                System.out.println("레코드가 없습니다.");
            }
            else {
                result.forEach(user ->
                    System.out.printf("| %s | %s | %tY-%<tm-%<td |\n",
                        user.getEmail(), user.getName(),
                        user.getCreateDate()));
            }
        }
        catch (Exception e) {
            e.printStackTrace();
            transaction.rollback();
        }

        //실행전 xml설정파일에서 none으로 변경
        em.close();
        emf.close();
    }
}
```

/B21aJPA_Basic/src/main/java/myjpa4/Use03_Parameter.java

none 인지 확인 후 테스트. 29라인의 검색어는 상황에 맞게 수정해볼것.

```
 1 package myjpa4;
 2 
 3⊕import java.util.List;
10 
11 public class Use03_Parameter {
12 
13⊖    public static void main(String[] args) {
14 
15         //none으로 변경한 후 실행
16         EntityManagerFactory emf =
17                 Persistence.createEntityManagerFactory("MyJPA");
18         EntityManager em = emf.createEntityManager();
19         EntityTransaction transaction = em.getTransaction();
20 
21         try {
22             transaction.begin();
23 
24             String SQL = "SELECT m FROM Member4 m "
25                     + " WHERE m.name = :name "
26                     + " ORDER BY m.name";
27             TypedQuery<Member4> query = em
28                     .createQuery(SQL, Member4.class)
29                     .setParameter("name", "양만춘");
30             List<Member4> result = query.getResultList();
31 
32             transaction.commit();
33 
34             if(result.isEmpty()) {
35                 System.out.println("레코드가 없습니다.");
36             }
37             else {
38                 result.forEach(user ->
39                     System.out.printf("| %s | %s | %tY-%<tm-%<td |\n",
40                         user.getEmail(), user.getName(),
41                         user.getCreateDate()));
42             }
43         }
44         catch (Exception e) {
45             e.printStackTrace();
46             transaction.rollback();
47         }
48 
49         em.close();
50         emf.close();
51     }
52 }
53 
```

/B21aJPA_Basic/src/main/java/myjpa4/Use04_Like.java

none 인지 확인 후 테스트.

```java
package myjpa4;

import java.util.List;

public class Use04_Like {

    public static void main(String[] args) {

        //none으로 변경한 후 실행
        EntityManagerFactory emf =
                Persistence.createEntityManagerFactory("MyJPA");
        EntityManager em = emf.createEntityManager();
        EntityTransaction transaction = em.getTransaction();

        try {
            transaction.begin();

            String SQL = "SELECT m FROM Member4 m "
                    + " WHERE m.email like :email "
                    + " ORDER BY m.name";
            TypedQuery<Member4> query = em
                    .createQuery(SQL, Member4.class)
                    .setParameter("email", "%spring.com%");
            List<Member4> result = query.getResultList();

            transaction.commit();

            if(result.isEmpty()) {
                System.out.println("레코드가 없습니다.");
            }
            else {

                result.forEach(user ->
                    System.out.printf("| %s | %s | %tY-%<tm-%<td |\n",
                        user.getEmail(), user.getName(),
                        user.getCreateDate()));
            }
        }
```

```java
        catch (Exception e) {
            e.printStackTrace();
            transaction.rollback();
        }

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