8-1. Mapped SuperClass - 매핑 정보 상속

그냥 진짜 단순하게 객체 입장에서 id, name이 너무 계속 나옴..
클래스 만들 때 마다 계속 만들기 귀찮으니까 부모 클래스에 이런걸 두고 멤버변수만 상속받아서 쓰고싶음.
DB는 완전히 다른데 객체 입장에서 멤버변수만 좀 돌려쓰고싶을 때 사용.

적용해보기

```
package hellojpa;

import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.EntityTransaction;
import javax.persistence.Persistence;
import java.time.LocalDateTime;
import java.util.ArrayList;
import java.util.List;

public class JpaMain {
  public static void main(String[] args) {
    EntityManagerFactory emf = Persistence.createEntityManagerFactory("hello");

    EntityManager em = emf.createEntityManager();

    EntityTransaction tx = em.getTransaction();
    tx.begin();
    try {
      Member member = new Member();
      member.setCreatedBy("Bong");
      member.setName("User1");
      member.setCreatedDate(LocalDateTime.now());

      em.flush(); // 영속성 컨텍스트에 있는 것을 db에 날리고
      em.clear(); // 영속성 컨텍스트에 있는 것을 모두 삭제하니까

      tx.commit();
```

```
    } catch (Exception e) {
      tx.rollback();
    } finally {
        em.close();
    }

    emf.close();

  }

}
```

```
package hellojpa;

import org.hibernate.annotations.ManyToAny;

import javax.persistence.*;
import java.time.LocalDateTime;
import java.util.ArrayList;
import java.util.Date;
import java.util.List;

@Entity
public class Member extends BaseEntity{

  @Id @GeneratedValue
  private Long id;

  @Column(name = "USERNAME")
  private String name;

//  @Column(name = "TEAM_ID")
//  private Long teamId;

  @ManyToOne// member 입장에서 team과는 다대일이므로 many to one
```

```
  @JoinColumn(name = "TEAM_ID")
  private Team team;

  @OneToOne
  @JoinColumn(name = "LOCKER_ID")
  private Locker locker;

  @ManyToMany
  @JoinTable(name = "MEMBER_PRODUCT")
  private List<Product> products = new ArrayList<>();

  @OneToMany(mappedBy = "member")
  private List<MemberProduct> memberProducts = new ArrayList<>();

  // 모든 파일에 들어가야 하는 것들... 을 DBA가 정해주면 적어주어야한다;; -> BaseEntity를 만들어 SuperClass 상속하기
  /*private String createdBy;
  private LocalDateTime createdDate;
  private String lastModifiedBy;
  private LocalDateTime lastModifiedDate;*/


  public Team getTeam() {
    return team;
  }

  public void setTeam(Team team) {
    this.team = team;
  }

  public Long getId() {
    return id;
  }

  public void setId(Long id) {
    this.id = id;
  }

  public String getName() {
```

```
    return name;
  }

  public void setName(String name) {
    this.name = name;
  }

  public void changeTeam(Team team) {
    this.team = team;
    team.getMembers().add(this);
  }
}
```

```
package hellojpa;

import javax.persistence.Entity;
import javax.persistence.MappedSuperclass;
import java.time.LocalDateTime;

@MappedSuperclass // mapping 정보만 받는 부모 클래스(슈퍼 클래스)
public class BaseEntity {

  private String createdBy;
  private LocalDateTime createdDate;
  private String lastModifiedBy;
  private LocalDateTime lastModifiedDate;

  public String getCreatedBy() {
    return createdBy;
  }

  public void setCreatedBy(String createdBy) {
    this.createdBy = createdBy;
  }
```

```
  public LocalDateTime getCreatedDate() {
    return createdDate;
  }

  public void setCreatedDate(LocalDateTime createdDate) {
    this.createdDate = createdDate;
  }

  public String getLastModifiedBy() {
    return lastModifiedBy;
  }

  public void setLastModifiedBy(String lastModifiedBy) {
    this.lastModifiedBy = lastModifiedBy;
  }

  public LocalDateTime getLastModifiedDate() {
    return lastModifiedDate;
  }

  public void setLastModifiedDate(LocalDateTime lastModifiedDate) {
    this.lastModifiedDate = lastModifiedDate;
  }
}
```

```
create table Member (
    id bigint not null,
     createdBy varchar(255),
     createdDate timestamp,
     lastModifiedBy varchar(255),
     lastModifiedDate timestamp,
     USERNAME varchar(255),
     LOCKER_ID bigint,
     TEAM_ID bigint,
     primary key (id)
)
```

나중에 event라는 기능으로 할 수 있고 spring을 도입하면 어노테이션으로도 할 수 있다.

주의
엔티티가 절대 아님! 그래서 상속관계 매핑도 안되고, 슈퍼클래스를 상속받는 자식클래스에 매핑 정보만 제공한다.

만약 BaseEntity를 단독으로 만들일이 없다면, 추상클래스로 만드는게 나음.

엔티티가 공통으로 사용하는 매핑 정보를 모으는 역할…

@Entity 클래스는 엔티티나 @MappedSuperclass로 지정한 클래스만 상속이 가능하다.
=> 상속을 받으려면 일단 상속받으려는 부모 클래스에 위 두 어노테이션 중 하나가 있어야함!!!!!

#JPA