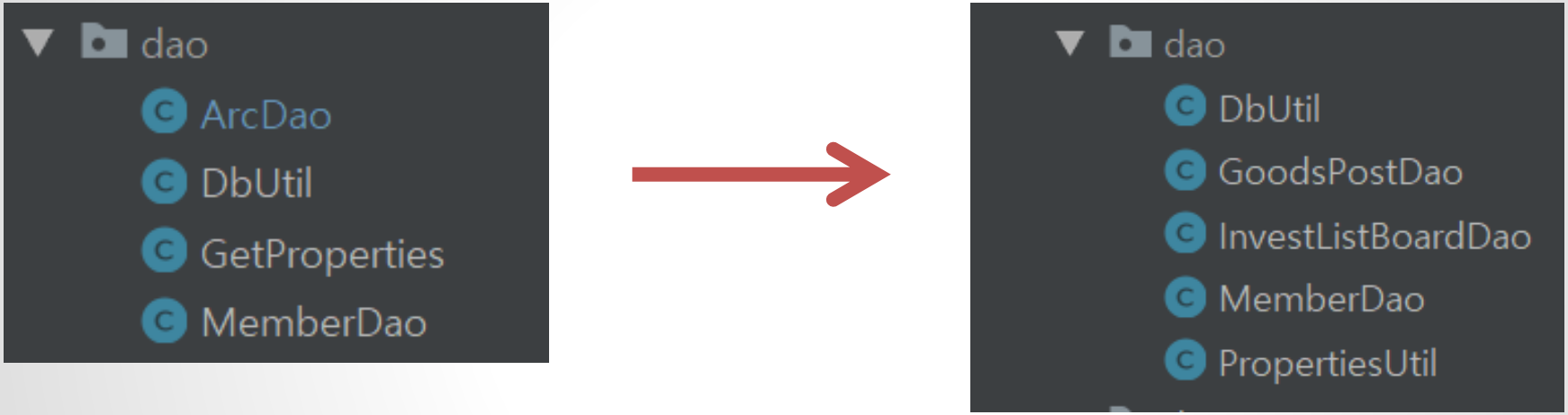
리팩토링 과정

DAO 클래스 명 변경



ArcDAO를 GoodsPostDao와 InvestLIstBoardDao로 나눠서 각 주제에 알맞게 세분화 (직관력 향상)

3

```
public ArcDao() {
    GetProperties gp = GetProperties.getInstance();
    dbURL = gp.getDbURL();
    properties = gp.getProperties();
}
```

```
public static Connection connect(String dbURL, Properties properties)
throws RuntimeException {
    Connection conn = null;
    try{
        Class.forName("com.mysql.cj.jdbc.Driver");
        conn = DriverManager.getConnection(dbURL, properties);
    }catch (Exception ex) {
        throw new RuntimeException(ex);
    }
    return conn;
}
```

```
public static Connection connect()
throws RuntimeException {
    Connection conn;
    try{
        PropertiesUtil propertiesUtil = PropertiesUtil.getInstance();
        Class.forName("com.mysql.cj.jdbc.Driver");
        conn = DriverManager.getConnection(propertiesUtil.getDbURL(),
                propertiesUtil.getProperties());
    }catch (Exception ex) {
        throw new RuntimeException(ex);
    }
    return conn;
}
```

ArcDao에서 Properties 불러오는 것을 connect에서 하게 수정
- 중복되는 코드 제거

Singletone 수정

```
private static PropertiesUtil instance;

public static PropertiesUtil getInstance(){
    if(instance==null){
        instance = new PropertiesUtil();
    }
    return instance;
}
```

```
private static PropertiesUtil instance = new PropertiesUtil();

public static PropertiesUtil getInstance(){
    return instance;
}
```

기존의 것은 두개의 쓰레드가 getInstance()를 동시에 실행하면 두개의 메모리가 생성되는 에러가 발생 할 수 도있다.

-> synchronized or instance 변수에서 인스턴스를 생성해준다.

1. SET @rownum 삭제
SET은 MySQL에서 지원 해주는 명령어인데 특정 DB에 종속적인것은 사용하지 않는게 좋다.
(LIMIT을 사용해서 해결)

2. FROM 안의 SELECT 절 삭제
FROM절 뒤에 SELECT가 들어가면 성능이 떨어질 수가 있다.
(while 안에 카운트를 주는 것으로 해결)

```
sql ="SET @rownum:=0;";
ps = conn.prepareStatement(sql);
rs = ps.executeQuery();

sql =   "SELECT A.*₩n" +
        "FROM (SELECT @rownum:=@rownum+1 AS ROW_NUM,₩n" +
        "m.gds_nm , gds.prf_rto, inv.inv_prod, inv.my_inv_prc, gds.cms₩n" +
        "FROM my_inv_lst inv, inv_gds_lst gds INNER JOIN gds_mst m ON gds.gds_cd = m.gds_cd₩n" +
        "WHERE inv.gds_cd = gds.gds_cd)A ₩n" +
        "WHERE A.ROW_NUM BETWEEN ? AND ?;";
```

```
sql =   "SELECT m.gds_nm , gds.prf_rto, inv.inv_prod, inv.my_inv_prc, gds.cms₩n" +
        "FROM my_inv_lst inv, inv_gds_lst gds INNER JOIN gds_mst m ON gds.gds_cd = m.gds_cd₩n" +
        "WHERE inv.gds_cd = gds.gds_cd LIMIT ?,?";
```

```
while(rs.next()) {
    MyGoodsListDto myGoodsListDto = new MyGoodsListDto();
    myGoodsListDto.setRownum(rs.getInt( columnIndex: 1));
    myGoodsListDto.setGoodsName(rs.getString( columnIndex: 2));
    myGoodsListDto.setPrfRto(rs.getLong( columnIndex: 3));
    myGoodsListDto.setInvestPeriod(rs.getInt( columnIndex: 4));
```

```
int rowNum=0;
while(rs.next()) {
    MyGoodsListDto myGoodsListDto = new MyGoodsListDto();
    myGoodsListDto.setRownum(rowNum++);
    myGoodsListDto.setGoodsName(rs.getString( columnLabel: "gds_nm"));
    myGoodsListDto.setPrfRto(rs.getLong( columnLabel: "prf_rto"));
    myGoodsListDto.setInvestPeriod(rs.getInt( columnLabel: "inv_prod"));
    myGoodsListDto.setMyPrice(rs.getInt( columnLabel: "A.my_inv_prc"));
```

columnIndex로 넘겨주는 것은 나중에 SQL문이 바뀔 경우 혹은 컬럼이 추가 될 경우에 다시 수정해줘야 하므로 효율적이지 못하므로 columnLabel을 직접 입력하는 방식으로 변경

DAO에 RuntimeException 추가

```
}catch (Exception ex) {
    ex.printStackTrace();
    throw new RuntimeException();
```

DAO에서 에러가 발생할 경우 사용자 입장에서는 에러가 발생했는지 정확히 알 수 가없으므로 RuntimeException을 추가해줘서 Servlet에서 에러창이 뜨게 추가.

```
}catch (Exception ex) {
    ex.printStackTrace();
}finally {
    DbUtil.close(conn, ps, rs);
}
return list;
```

```
try {
    list = investListBoardDao.getMyGoodsList(pg, posts);
}catch (RuntimeException re){
    PrintWriter out = resp.getWriter();
    out.println("<script language='javascript'>");
    out.println("alert('Error');");
    out.println("window.location.href = ₩"/₩";");
    out.println("</script>");
    out.close();
}
```

```java
package my.examples.arc.dto;

public class ARCGdsMstDto {
    private int gds_cd;
    private String gds_nm;

    public int getGds_cd() { return gds_cd; }

    public void setGds_cd(int gds_cd) { this.gds_cd = gds_cd; }

    public String getGds_nm() { return gds_nm; }

    public void setGds_nm(String gds_nm) { this.gds_nm = gds_nm; }
}
```

```java
package my.examples.arc.dto;

public class ARCGdsMstDto {
    private int gdsCd;
    private String gdsNm;

    public int getGdsCd() {
        return gdsCd;
    }

    public void setGdsCd(int gdsCd) {
        this.gdsCd = gdsCd;
    }

    public String getGdsNm() {
        return gdsNm;
    }
```

JAVA Coding Convention(Camel Toe)에 어긋나는 변수 명을 일관성 있게 통일 함.