Mobile Application.

1). Src.
2). Resources
3). test
4). JRE System Library
5). Referenced Libraries.
6). lib.

Src.

1). Bean.
2). Dao
3). Exception.
4). pi
5). Service.
6). util.

entities = classes

Bean.

1). MobileBean Class
- → private variables.
- → Default Constructor.
- → parameterized Constructor.
- → getters and Setters.
- → toString method.

Now private variables. names.

int → mobileId, name, price, quantity ← String
↑ float

2). PurchaseDetails ~~Class~~ Bean Class

→ Define above things

Now private variables names

int → purchaseId, name, mailId, phoneNO, purDate, mobileId.
↑ string (phoneNO)
↑ LocalDate (purDate)
↑ int (mobileId)

Dao.

A) Two interfaces
B) Two classes.
c). Two querry mapper interfaces.

1). Interface IMobileDAO

a). boolean updateMobile (int mobileId, int quantity)

b). List < MobileBean> viewAll()
c). List < MobileBean> search (float minPrice, float maxPrice)

d). int getQuantity (int mobileId)

All above methods throws MobilePurchase Exception

2). Interface IPurchaseDetailsDAO
boolean
a). insertPurchase ~~Details~~ (PurchaseDetailsBean purchaseDetailsBean)
b). boolean deletePurchaseDetails (int mobileId)

All above methods throws MobilePurchase Exception

NOTE: In above two interfaces make all arguments final.

Exception. inherited × ⟷ browse → Exception.

class MobilePurchaseException extends Exception

1). Default constructor. → source → super string ②.

2). Parameterized Constructor (String message)
{ super(message);
}

→ source → super

3). serialVersionUID= . . .

CTRL ① change name. → exception. ① throwable

DAO: ② ctrl+shift+O.

3). class MobileDAOImpl implements IMobileDAO
add → interface → inherit, abstract

(a). ~~Constructor~~

```
public boolean updateMobile (int mobileId,
    int quantity) throws MobilePurchaseException.
{1) → int records = 0;
 2) → boolean isUpdated = false;
 3) → try (Connection connMobile
            = DBConnection.getInstance().
                getConnection();
      PreparedStatement preparedStatement
          = connMobile.preparedStatement
            (QueryMapperMobile.UPDATE_MOBILES);
      )
 4) → { preparedStatement.setString (1, Integer.
                        toString (quantity));
      preparedStatement.setInt (2, mobileId);
```

```
5)→ records = preparedStatement. executeUpdate();
6)→ if ( records >0)
    {
      isUpdated = true;
    }
7)→ }
8)→ catch ( SQLException sqlEx)
    {
     throw new MobilePurchaseException
              ( sqlEx . getMessage ());
    }
9)→ return isUpdated;
10) → }
```

(b) Similarly for deleteMobile (int MobileId)

```
(c). List <MobileBean> viewAll () throws
        MobilePurchaseException
   {
1)→ List <MobileBean> mobileList = new
            ArrayList <MobileBean> ();

2)→try { (.....);
        ResultSet rsMobiles = preparedStatement
                    . executeQuerry ();)
3)→ {
3)→ while ( rsMobiles. next())
    {
4)→MobileBean mobile = new MobileBean();
```

```
5)→ mobile. setMobileId (rsMobiles • getInt ("mobileId")
                                                        );
6)→ Similarly , for name , price , Quantity .
 7)→  mobileList • add (mobile);
 8)→ }
 9)→  if (mobileList • size()==0)
      { throw new MobilePurchaseException
             ("No records found");
        }
10)→ }
11)→  catch ( ... )
      { . . .
        }
12)→  return  mobileList ;
13) → }
(d). List < MobileBean> search ( float minPrice,
           float maxPrice) . . .
     ~~Add~~ Similar.
    { preparedStatement . setFloat (1, minPrice);
        . . . .
       while ( . . .)
       { . . mobile. setPrice (rsMobiles. getFloat
                                  ("price"));
         . . . .
         }
```

```
(c). int getQuantity (int MobileId)...
 { int mobileQty = 0;
   try(...);
    {
       ....
     if ( rsMobiles.next())
     {
       mobileQty = Integer.parseInt
             (rsMobiles.getString ("quantity"));
      }
    }
   catch(...)
    {...;
    }
   return mobileQty;
  }
}
```

```
class
4). PurchaseDetailsDAOImpl implements
        IPurchaseDetailsDAO

 {
a)> boolean insertPurchase (.....)
      {
        int records = 0;
        boolean isInserted = false;
                connPurchaseDetails
     try ( ....(QueryMapperPurchaseDetails.
                        INSERT_PURCHASE);
         )

  {
     java.sql.Date purchaseDate
       = new Date (new java.util.Date().
                          getTime());
  preparedStatement .setString (1,
         purchaseDetailsBean .getName());
  preparedStatement .setDate (2, purchaseDate);
   ....
   ...
   ...

b)-> boolean deletePurchaseDetails (int MobileId)
```

5). Interface QueryMapperMobile
{ public static final String UPDATE_MOBILES
a)→ = "UPDATE mobiles SET quantity = ?
WHERE mobileId = ?" ;

b)→
c)→
d)→
}

6). Interface Q.M.PD
{
. INSERT_PURCHASE
= "INSERT INTO purchasedetails
VALUES (purchMobile_sequence.NEXTVAL,
?, ?, ?, ?, ?)";

Service

~~Interface~~
1). IServiceMobile add → interface
{ viewAll() inherit → abstract
deleteMobile(int mobileId)
search (float minPrice, float maxPrice)

2). Interface IServicePurchaseMobile
{ insertPurchaseDetails.

```
3). ServiceMobileImple implements IServiceMobile
{
a)→ IMobileDAO mobileDAO;
b)→   public ServiceImple()
   {   mobileDAO = new MobileDAOImpl();
   }
c)→ List<MobileBean> viewAll() throws
    MobilePurchaseException
    {
     List<MobileBean> mobileList = mobileDAO
                                   viewAll();
     return mobileList;
    }
d)→ boolean deleteMobile (int mobileId)
     throws ...
 {
  IPurchaseDetailsDAO purchaseDetailsDAO  —ⓐ
       = new PurchaseDetailsDAOImpl();
  boolean isPurchaseDeleted = p....       —ⓑ
        deletePurchaseDetails (mobileId);
  boolean isDeleted = mobileDAO. deleteMobile
                          (mobileId);
     return ("11");
  }
e)→ list search . . .
```

4). ServiePurchaseImpl implements . . .

```
int mobileQuantity = 0;
boolean isItInserted = false;
boolean isUpdated = false;
(a):
(b):
mobileQuantity = mobileDAO.getQuantity
              (purchaseDetailsBean.getMobileId());
 if (mobileQuantity >0)
 {
 isItinserted = purchaseDetailsDAO.
                insertPurchase(purchaseDetails
                                Bean)
   mobileQuantity--;
   isUpdated = mobileDAO.updateMobile
               (purchaseDetailsBean.getMobileId
                (), mobileQuantity);
 }
 return (isItinserted && isUpdated)
```

Name
Phone
Mail
MobileId.

Packages

(1). Bean package

(A) (a). create class → MobileBean.

(b). Look on database table i.e. mobileId....
quantity.

(c). Write parameters depending which
data given in database table.

(d). Then create one default and one
parameteriesed constructor.

(e). Then create getters and setters.

(f). Then create toString().

→ these are created by clicking on
source then select appropriate functions.

(2) (B). Create class PurchaseDetails..
Repeat all steps above..

(2). Dao Package

(1) Create Interface PurchaseDetailsDAO

(2). Check what parameters we required
through question statement.

(3). Create another Interface for mobile.

(4). check parameters.

~~(5)~~ check package should be util.

(6) Create another Interface for MapperQuery for
purchase

~~(7)~~. Make variables final. (MapperQuery)

(8). Create another Interface for Mobile

(9). In MapperQuery interface we write

[illegible]

Libraries..

(1). Create project.
(2). Create packages. (same).
(3). ~~Copy resource, test, files from donor~~ ~~application~~

(3) Now click right click on project → go to → new → create new source folder.
(Resources and test)

(4). Inside resources copy files from donor application.

(5) Now create lib folder (file folder)

(6). Now copy files from lib folder of donorapplication to lib folder to mobile.

(7). Now build path of those files which copied to lib. folder.

(8) After building path, new folder created named reference libraries.

(9) Now copy all tables from database.

(10) Change username & password (Jdbc).

Now. Packages.

(1) util Package.
(a). copy file from donor (replace names)
↓
Jdbc.properties
(b) import packages.

(2) Exception package.
(a). extends exception
(b) add serial...
(c). Constructor.