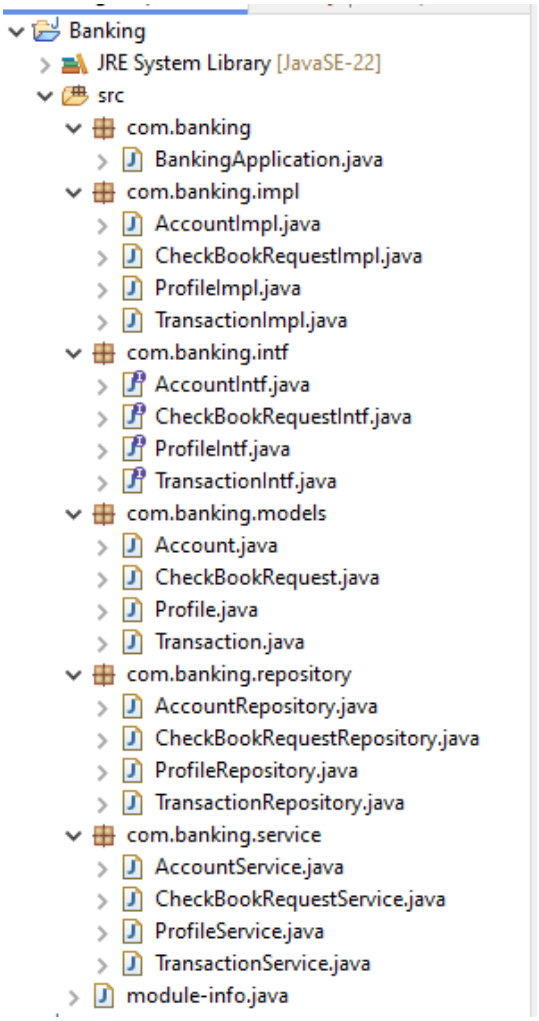
JAVA PROJECT

NET BANKING

- **Profiles:** The system displays the profile details, updates one profile, and shows the remaining profiles after an update.
- **Accounts:** The system adds an account, updates the balance, and then shows the remaining accounts after an update.
- **Transactions:** The system shows the initial transaction, updates one, deletes another, and displays the remaining transactions.
- **CheckBook Requests:** The system manages checkbook requests by showing the initial status, updating one, and displaying the remaining request after deletion.

Project structure:



Profile Module:

Profile.java :

```
package com.banking.models;
/**
 * Represents a user profile in the banking system.
 */
public class Profile {
   private String profileId;
   private String name;
   private String email;
   // Constructor
   public Profile(String profileId, String name, String email) {
      this.profileId = profileId;
      this.name = name;
      this.email = email;
   }
   // Getters and Setters
   public String getProfileId() {
      return profileId;
   }
   public void setProfileId(String profileId) {
      this.profileId = profileId;
   }
   public String getName() {
      return name;
   }
   public void setName(String name) {
      this.name = name;
   }
   public String getEmail() {
      return email;
   }
   public void setEmail(String email) {
      this.email = email;
   }
   @Override
   public String toString() {
      return "Profile{" +
            "profileId='" + profileId + '\'' +
            ", name='" + name + '\'' +
            ", email='" + email + '\'' +
            '}';
   }
}
```

ProfileIntf.java :

```
package com.banking.intf;
import com.banking.models.Profile;
/**
 * Interface for Profile operations.
 * Defines methods for CRUD operations related to Profile.
 */
public interface ProfileIntf {
   void createProfile(Profile profile);
   Profile getProfileById(String profileId);
   void updateProfile(Profile profile);
   void deleteProfile(String profileId);
}
```

ProfileService.java :

```
package com.banking.service;
import com.banking.intf.ProfileIntf;
import com.banking.models.Profile;
/**
 * Service class for managing profiles.
 * Handles business logic related to Profile operations.
 */
public class ProfileService {
   private ProfileIntf profileIntf;
   public ProfileService(ProfileIntf profileIntf) {
      this.profileIntf = profileIntf;
   }
   public void createProfile(Profile profile) {
      profileIntf.createProfile(profile);
   }
   public Profile getProfileById(String profileId) {
      return profileIntf.getProfileById(profileId);
   }
   public void updateProfile(Profile profile) {
      profileIntf.updateProfile(profile);
   }
   public void deleteProfile(String profileId) {
      profileIntf.deleteProfile(profileId);
   }
}
```

ProfileRepository.java :

```
package com.banking.repository;
import com.banking.models.Profile;
import java.util.*;
/**
 * Repository for managing Profile data.
 */
public class ProfileRepository {
   private Map<String, Profile> profileMap = new HashMap<>();
   // Adds a new profile to the repository
   public void addProfile(Profile profile) {
      profileMap.put(profile.getProfileId(), profile);
   }
   // Retrieves a profile by its ID
   public Profile getProfileById(String profileId) {
      return profileMap.get(profileId);
   }
   // Updates an existing profile
   public void updateProfile(Profile profile) {
      profileMap.put(profile.getProfileId(), profile);
   }
   // Deletes a profile by its ID
   public void deleteProfile(String profileId) {
      profileMap.remove(profileId);
   }
   // Retrieves all profiles
   public Collection<Profile> getAllProfiles() {
      return profileMap.values();
   }
}
```

ProfileImpl.java :

```
package com.banking.impl;
import com.banking.intf.ProfileIntf;
import com.banking.models.Profile;
import com.banking.repository.ProfileRepository;
/**
 * Implementation of ProfileIntf using ProfileRepository for data storage.
 */
public class ProfileImpl implements ProfileIntf {
   private ProfileRepository profileRepository = new ProfileRepository();
   @Override
   public void createProfile(Profile profile) {
      profileRepository.addProfile(profile);
   }
```

```
    @Override
    public Profile getProfileById(String profileId) {
        return profileRepository.getProfileById(profileId);
    }
    @Override
    public void updateProfile(Profile profile) {
        profileRepository.updateProfile(profile);
    }
    @Override
    public void deleteProfile(String profileId) {
        profileRepository.deleteProfile(profileId);
    }
}
```

Account Module:

- **Account.java**: The Account model class defines the structure of an account, including accountId, accountType, and balance.
- **AccountIntf.java**: The AccountIntf interface declares methods for managing accounts, including adding, retrieving, updating, and deleting accounts.
- **AccountImpl.java**: The AccountImpl class implements the AccountIntf interface and provides the actual logic for managing accounts using a HashMap.
- **AccountRepository.java**: The AccountRepository class returns sample account data that can be used to populate the system.
- **AccountService.java**: The AccountService class demonstrates how to use the AccountImpl class to manage accounts, including adding, reading, updating, and deleting accounts.

Account.java :

```
package com.banking.models;
/**
 * Represents a bank account.
 */
public class Account {
    private String accountId;
    private String accountType;
    private String profileId;
    // Constructor
    public Account(String accountId, String accountType, String profileId) {
        this.accountId = accountId;
        this.accountType = accountType;
        this.profileId = profileId;
```

```
    }
    // Getters and Setters
    public String getAccountId() {
        return accountId;
    }
    public void setAccountId(String accountId) {
        this.accountId = accountId;
    }
    public String getAccountType() {
        return accountType;
    }
    public void setAccountType(String accountType) {
        this.accountType = accountType;
    }
    public String getProfileId() {
        return profileId;
    }
    public void setProfileId(String profileId) {
        this.profileId = profileId;
    }
    @Override
    public String toString() {
        return "Account{" +
                "accountId='" + accountId + '\'' +
                ", accountType='" + accountType + '\'' +
                ", profileId='" + profileId + '\'' +
                '}';
    }
}
```

AccountIntf.java :

```
package com.banking.intf;
import com.banking.models.Account;
/**
 * Interface for Account operations.
 * Defines methods for CRUD operations related to Account.
 */
public interface AccountIntf {
    void createAccount(Account account);
    Account getAccountById(String accountId);
    void updateAccount(Account account);
    void deleteAccount(String accountId);
}
```

AccountImpl.java :

```
package com.banking.impl;
import com.banking.intf.AccountIntf;
import com.banking.models.Account;
import com.banking.repository.AccountRepository;
/**
 * Implementation of AccountIntf using AccountRepository for data storage.
 */
public class AccountImpl implements AccountIntf {
   private AccountRepository accountRepository = new AccountRepository();
   @Override
   public void createAccount(Account account) {
      accountRepository.addAccount(account);
   }
   @Override
   public Account getAccountById(String accountId) {
      return accountRepository.getAccountById(accountId);
   }
   @Override
   public void updateAccount(Account account) {
      accountRepository.updateAccount(account);
   }
   @Override
   public void deleteAccount(String accountId) {
      accountRepository.deleteAccount(accountId);
   }
}
```

AccountRepository.java :

```
package com.banking.repository;
import com.banking.models.Account;
import java.util.*;
/**
 * Repository for managing Account data.
 */
public class AccountRepository {
   private Map<String, Account> accountMap = new HashMap<>();
   // Adds a new account to the repository
   public void addAccount(Account account) {
      accountMap.put(account.getAccountId(), account);
   }
   // Retrieves an account by its ID
   public Account getAccountById(String accountId) {
      return accountMap.get(accountId);
   }
```

```
    // Updates an existing account
    public void updateAccount(Account account) {
        accountMap.put(account.getAccountId(), account);
    }
    // Deletes an account by its ID
    public void deleteAccount(String accountId) {
        accountMap.remove(accountId);
    }
    // Retrieves all accounts
    public Collection<Account> getAllAccounts() {
        return accountMap.values();
    }
}
```

AccountService.java :

```
package com.banking.service;
import com.banking.intf.AccountIntf;
import com.banking.models.Account;
/**
 * Service class for managing accounts.
 * Handles business logic related to Account operations.
 */
public class AccountService {
    private AccountIntf accountIntf;
    public AccountService(AccountIntf accountIntf) {
        this.accountIntf = accountIntf;
    }
    public void createAccount(Account account) {
        accountIntf.createAccount(account);
    }
    public Account getAccountById(String accountId) {
        return accountIntf.getAccountById(accountId);
    }
    public void updateAccount(Account account) {
        accountIntf.updateAccount(account);
    }
    public void deleteAccount(String accountId) {
        accountIntf.deleteAccount(accountId);
    }
}
```

Transaction Module:

- **Transaction.java**: The Transaction model class defines the structure of a transaction, including transactionId, accountId, amount, type (debit/credit), and date.
- **TransactionIntf.java**: The TransactionIntf interface declares methods for managing transactions, including adding, retrieving, updating, and deleting transactions.
- **TransactionImpl.java**: The TransactionImpl class implements the TransactionIntf interface and provides the actual logic for managing transactions using a HashMap.
- **TransactionRepository.java**: The TransactionRepository class returns sample transaction data that can be used to populate the system.
- **TransactionService.java**: The TransactionService class demonstrates how to use the TransactionImpl class to manage transactions, including adding, reading, updating, and deleting transactions.

Transaction.java:

```
package com.banking.models;
/**
 * Represents a transaction in the banking system.
 */
public class Transaction {
   private String transactionId;
   private String accountId;
   private double amount;
   private String type;
   // Constructor
   public Transaction(String transactionId, String accountId, double amount, String type) {
      this.transactionId = transactionId;
      this.accountId = accountId;
      this.amount = amount;
      this.type = type;
   }
   // Getters and Setters
   public String getTransactionId() {
      return transactionId;
   }
   public void setTransactionId(String transactionId) {
      this.transactionId = transactionId;
   }

   public String getAccountId() {
      return accountId;
   }
   public void setAccountId(String accountId) {
```

```
        this.accountId = accountId;
    }
    public double getAmount() {
        return amount;
    }
    public void setAmount(double amount) {
        this.amount = amount;
    }
    public String getType() {
        return type;
    }
    public void setType(String type) {
        this.type = type;
    }
    @Override
    public String toString() {
        return "Transaction{" +
                "transactionId='" + transactionId + '\'' +
                ", accountId='" + accountId + '\'' +
                ", amount=" + amount +
                ", type='" + type + '\'' +
                '}';
    }
}
```

TransactionIntf.java:

```
package com.banking.intf;
import com.banking.models.Transaction;
/**
 * Interface for Transaction operations.
 * Defines methods for CRUD operations related to Transaction.
 */
public interface TransactionIntf {
    void createTransaction(Transaction transaction);
    Transaction getTransactionById(String transactionId);
    void updateTransaction(Transaction transaction);
    void deleteTransaction(String transactionId);
}
```

TransactionImpl.java:

```
package com.banking.impl;
import com.banking.intf.TransactionIntf;
import com.banking.models.Transaction;
import com.banking.repository.TransactionRepository;
/**
 * Implementation of TransactionIntf using TransactionRepository for data storage.
 */
public class TransactionImpl implements TransactionIntf {
```

```
    private TransactionRepository transactionRepository = new TransactionRepository();
    @Override
    public void createTransaction(Transaction transaction) {
        transactionRepository.addTransaction(transaction);
    }
    @Override
    public Transaction getTransactionById(String transactionId) {
        return transactionRepository.getTransactionById(transactionId);
    }
    @Override
    public void updateTransaction(Transaction transaction) {
        transactionRepository.updateTransaction(transaction);
    }
    @Override
    public void deleteTransaction(String transactionId) {
        transactionRepository.deleteTransaction(transactionId);
    }
}
```

TransactionRepository.java:

```
package com.banking.repository;
import com.banking.models.Transaction;
import java.util.*;
/**
 * Repository for managing Transaction data.
 */
public class TransactionRepository {
    private Map<String, Transaction> transactionMap = new HashMap<>();
    // Adds a new transaction to the repository
    public void addTransaction(Transaction transaction) {
        transactionMap.put(transaction.getTransactionId(), transaction);
    }
    // Retrieves a transaction by its ID
    public Transaction getTransactionById(String transactionId) {
        return transactionMap.get(transactionId);
    }
    // Updates an existing transaction
    public void updateTransaction(Transaction transaction) {
        transactionMap.put(transaction.getTransactionId(), transaction);
    }
    // Deletes a transaction by its ID
    public void deleteTransaction(String transactionId) {
        transactionMap.remove(transactionId);
    }
    // Retrieves all transactions
    public Collection<Transaction> getAllTransactions() {
        return transactionMap.values();
    }
}
```

TransactionService.java:

```
package com.banking.service;
import com.banking.intf.TransactionIntf;
import com.banking.models.Transaction;
/**
 * Service class for managing transactions.
 * Handles business logic related to Transaction operations.
 */
public class TransactionService {
   private TransactionIntf transactionIntf;
   public TransactionService(TransactionIntf transactionIntf) {
      this.transactionIntf = transactionIntf;
   }
   public void createTransaction(Transaction transaction) {
      transactionIntf.createTransaction(transaction);
   }
   public Transaction getTransactionById(String transactionId) {
      return transactionIntf.getTransactionById(transactionId);
   }
   public void updateTransaction(Transaction transaction) {
      transactionIntf.updateTransaction(transaction);
   }
   public void deleteTransaction(String transactionId) {
      transactionIntf.deleteTransaction(transactionId);
   }
}
```

CheckBookRequest Module:

- **CheckBookRequest.java**: The CheckBookRequest model class defines the structure of a checkbook request, including requestId, accountId, numberOfLeaves, requestDate, and status.
- **CheckBookRequestIntf.java**: The CheckBookRequestIntf interface declares methods for managing checkbook requests, including adding, retrieving, updating, and deleting checkbook requests.
- **CheckBookRequestImpl.java**: The CheckBookRequestImpl class implements the CheckBookRequestIntf interface and provides the actual logic for managing checkbook requests using a HashMap.
- **CheckBookRequestRepository.java**: The CheckBookRequestRepository class returns sample checkbook request data that can be used to populate the system.
- **CheckBookRequestService.java**: The CheckBookRequestService class demonstrates how to use the CheckBookRequestImpl class to manage checkbook requests, including adding, reading, updating, and deleting requests.

CheckBookRequest.java :

```
package com.banking.models;
/**
 * Represents a request for a checkbook.
 */
public class CheckBookRequest {
   private String requestId;
   private String accountId;
   private String requestDate;
   // Constructor
   public CheckBookRequest(String requestId, String accountId, String requestDate) {
      this.requestId = requestId;
      this.accountId = accountId;
      this.requestDate = requestDate;
   }
   // Getters and Setters
   public String getRequestId() {
      return requestId;
   }
   public void setRequestId(String requestId) {
      this.requestId = requestId;
   }
   public String getAccountId() {
      return accountId;
   }
   public void setAccountId(String accountId) {
      this.accountId = accountId;
   }
   public String getRequestDate() {
      return requestDate;
   }
   public void setRequestDate(String requestDate) {
      this.requestDate = requestDate;
   }
   @Override
   public String toString() {
      return "CheckBookRequest{" +
            "requestId='" + requestId + '\'' +
            ", accountId='" + accountId + '\'' +
            ", requestDate='" + requestDate + '\'' +
            '}';
   }
}
```

CheckBookRequestIntf.java:

```
package com.banking.intf;
import com.banking.models.CheckBookRequest;
/**
 * Interface for CheckBookRequest operations.
 * Defines methods for CRUD operations related to CheckBookRequest.
 */
public interface CheckBookRequestIntf {
   void createCheckBookRequest(CheckBookRequest request);
   CheckBookRequest getCheckBookRequestById(String requestId);
   void updateCheckBookRequest(CheckBookRequest request);
   void deleteCheckBookRequest(String requestId);
}
```

CheckBookRequestImpl.java:

```
package com.banking.impl;
import com.banking.intf.CheckBookRequestIntf;
import com.banking.models.CheckBookRequest;
import com.banking.repository.CheckBookRequestRepository;
/**
 * Implementation of CheckBookRequestIntf using CheckBookRequestRepository for data
storage.
 */
public class CheckBookRequestImpl implements CheckBookRequestIntf {
   private CheckBookRequestRepository checkBookRequestRepository = new
CheckBookRequestRepository();
   @Override
   public void createCheckBookRequest(CheckBookRequest request) {
      checkBookRequestRepository.addCheckBookRequest(request);
   }
   @Override
   public CheckBookRequest getCheckBookRequestById(String requestId) {
      return checkBookRequestRepository.getCheckBookRequestById(requestId);
   }
   @Override
   public void updateCheckBookRequest(CheckBookRequest request) {
      checkBookRequestRepository.updateCheckBookRequest(request);
   }
   @Override
   public void deleteCheckBookRequest(String requestId) {
      checkBookRequestRepository.deleteCheckBookRequest(requestId);
   }
}
```

CheckBookRequestRepository.java:

```
package com.banking.repository;
import com.banking.models.CheckBookRequest;
import java.util.*;
/**
 * Repository for managing CheckBookRequest data.
 */
public class CheckBookRequestRepository {
   private Map<String, CheckBookRequest> checkBookRequestMap = new HashMap<>();
   // Adds a new check book request to the repository
   public void addCheckBookRequest(CheckBookRequest request) {
      checkBookRequestMap.put(request.getRequestId(), request);
   }
   // Retrieves a check book request by its ID
   public CheckBookRequest getCheckBookRequestById(String requestId) {
      return checkBookRequestMap.get(requestId);
   }
   // Updates an existing check book request
   public void updateCheckBookRequest(CheckBookRequest request) {
      checkBookRequestMap.put(request.getRequestId(), request);
   }
   // Deletes a check book request by its ID
   public void deleteCheckBookRequest(String requestId) {
      checkBookRequestMap.remove(requestId);
   }
   // Retrieves all check book requests
   public Collection<CheckBookRequest> getAllCheckBookRequests() {
      return checkBookRequestMap.values();
   }
}
```

CheckBookRequestService.java:

```
package com.banking.service;
import com.banking.intf.CheckBookRequestIntf;
import com.banking.models.CheckBookRequest;
/**
 * Service class for managing check book requests.
 * Handles business logic related to CheckBookRequest operations.
 */
public class CheckBookRequestService {
   private CheckBookRequestIntf checkBookRequestIntf;
   public CheckBookRequestService(CheckBookRequestIntf checkBookRequestIntf) {
      this.checkBookRequestIntf = checkBookRequestIntf;
   }

   public void createCheckBookRequest(CheckBookRequest request) {
```

```
        checkBookRequestIntf.createCheckBookRequest(request);
    }
    public CheckBookRequest getCheckBookRequestById(String requestId) {
        return checkBookRequestIntf.getCheckBookRequestById(requestId);
    }
    public void updateCheckBookRequest(CheckBookRequest request) {
        checkBookRequestIntf.updateCheckBookRequest(request);
    }
    public void deleteCheckBookRequest(String requestId) {
        checkBookRequestIntf.deleteCheckBookRequest(requestId);
    }
}
```

Main Method (**BankingApplication.java**) :

```
package com.banking;
import com.banking.models.*;
import com.banking.repository.*;

/**
 * Main class for the Banking Application.
 * <p>
 * This application demonstrates how to use repositories to manage data for
 * Profile, Account, Transaction, and CheckBookRequest entities. It performs
 * CRUD operations (Create, Read, Update, Delete) and displays the results.
 * </p>
 */
public class BankingApplication {

    /**
     * Main method that executes the application logic.
     * <p>
     * It creates instances of repository classes, performs CRUD operations on
     * each type of entity (Profile, Account, Transaction, CheckBookRequest),
     * and prints the results to the console.
     * </p>
     *
     * @param args command-line arguments (not used)
     */
    public static void main(String[] args) {
        // Create repository instances
        ProfileRepository profileRepo = new ProfileRepository();
        AccountRepository accountRepo = new AccountRepository();
        TransactionRepository transactionRepo = new TransactionRepository();
        CheckBookRequestRepository checkBookRequestRepo = new
CheckBookRequestRepository();

        // Create new profiles
        Profile profile1 = new Profile("1", "John Doe", "john.doe@example.com");
        Profile profile2 = new Profile("2", "Jane Smith", "jane.smith@example.com");
```

```
profileRepo.addProfile(profile1);
profileRepo.addProfile(profile2);

// Create new accounts
Account account1 = new Account("A1", "Savings", "1");
Account account2 = new Account("A2", "Checking", "2");
accountRepo.addAccount(account1);
accountRepo.addAccount(account2);

// Create new transactions
Transaction transaction1 = new Transaction("T1", "A1", 100.00, "Deposit");
Transaction transaction2 = new Transaction("T2", "A2", 50.00, "Withdrawal");
transactionRepo.addTransaction(transaction1);
transactionRepo.addTransaction(transaction2);

// Create new check book requests
CheckBookRequest request1 = new CheckBookRequest("R1", "A1", "2024-08-15");
CheckBookRequest request2 = new CheckBookRequest("R2", "A2", "2024-08-16");
checkBookRequestRepo.addCheckBookRequest(request1);
checkBookRequestRepo.addCheckBookRequest(request2);

// Display all profiles
System.out.println("All Profiles:");
profileRepo.getAllProfiles().forEach(System.out::println);

// Display all accounts
System.out.println("\nAll Accounts:");
accountRepo.getAllAccounts().forEach(System.out::println);

// Display all transactions
System.out.println("\nAll Transactions:");
transactionRepo.getAllTransactions().forEach(System.out::println);

// Display all check book requests
System.out.println("\nAll Check Book Requests:");
checkBookRequestRepo.getAllCheckBookRequests().forEach(System.out::println);

// Update profile
Profile updatedProfile1 = new Profile("1", "John Doe", "john.new@example.com");
profileRepo.updateProfile(updatedProfile1);
System.out.println("\nUpdated Profile with ID 1: " + profileRepo.getProfileById("1"));

// Delete profile
profileRepo.deleteProfile("2");
System.out.println("\nProfiles after deletion:");
profileRepo.getAllProfiles().forEach(System.out::println);

// Delete account
accountRepo.deleteAccount("A2");
```

```
        System.out.println("\nAccounts after deletion:");
        accountRepo.getAllAccounts().forEach(System.out::println);

        // Update transaction
        Transaction updatedTransaction1 = new Transaction("T1", "A1", 200.00, "Deposit");
        transactionRepo.updateTransaction(updatedTransaction1);
        System.out.println("\nUpdated Transaction with ID T1: " +
transactionRepo.getTransactionById("T1"));

        // Delete check book request
        checkBookRequestRepo.deleteCheckBookRequest("R2");
        System.out.println("\nCheck Book Requests after deletion:");
        checkBookRequestRepo.getAllCheckBookRequests().forEach(System.out::println);
    }
}
```

OUTPUT: