1 Account.java

```java
package socialmedia;

import java.io.Serializable;
import java.util.ArrayList;
import java.util.List;

/**
 * This class stores information of each Account in the social media platform.
 * Accounts can be created or removed and their description field can be updated.
 * This class is a superclass of Posts and Comments classes.
 * @author Aisha
 * @author Saida
 *
 */
@SuppressWarnings("serial")
public class Account implements Serializable {

   //declaring variables
    private int ID;
    private String handle;
    private String description;
    private static int lastAccountID = 0;
    private static ArrayList<Account> accounts = new ArrayList<>();
    private List<Post> postsList = new ArrayList<>();
    private List<Endorsement> accountEndorsementsList = new ArrayList<>();
    private List<Comment> accountCommentsList = new ArrayList<>();

    /**
     * constructor for objects of class Account.
     * @param handle
     * @param description
     */
    public Account(String handle, String description) {
        this.handle = handle;
        this.description = description;
        this.ID = ++lastAccountID;
    }

    /**
     * one-arg constructor for objects of class Account.
     * @param handle
     */
    public Account(String handle) {
      this.handle = handle;
      this.ID = ++lastAccountID;
    }

    //setter for description
    public void setDescription(String description) {
      this.description = description;
    }

    //setter for handle
    public void setHandle(String handle) {
      this.handle = handle;
    }

    //getter for handle
    public String getHandle() {
        return handle;
    }

    //getter for the static list of accounts
    public static ArrayList<Account> getAccountsList() {
      return accounts;
    }

    //getter for description
    public String getDescription() {
        return description;
    }

    //getter for the list of posts of each account
    public List<Post> getEachPostList() {
      return postsList;
    }

    //getter for the list of endorsements of each account
    public List<Endorsement> getAccountEndorsementsList() {
      return accountEndorsementsList;
    }

    //getter for the list of comments of each account
    public List<Comment> getAccountCommentList() {
      return accountCommentsList;
    }

    /**
     * This method adds an account into the static accounts list.
     */
    public static void addAccount(Account account){
        accounts.add(account);
    }

    /**
     * This method adds an endorsement to the endorsement list of each account.
     */
    public void addAccountEndorsement(Endorsement endorsement) {
        accountEndorsementsList.add(endorsement);
    }

    /**
     * This method returns the ID of the Account.
     * @return ID
     */
    public int getID(){
        return ID;
    }

    /**
     * This method displays the Account.
     * @return toString of Account
     */
    public String toString() {
        return "Account[name = " + handle + ", description = " +
        description + ", ID = " + ID + "]";
    }
}
```

2 Comment.java

```
package socialmedia;
import java.util.ArrayList;
import java.util.List;

/**
 * This class stores information on each Endorsement in the social media platform.
 * Endorsements automatically replicate the endorsed message. The endorsed message cannot be endorsed again.
 * This class is a subclass of Post class.
 * @author Aisha
 * @author Saida
 */
@SuppressWarnings("serial")
public class Comment extends Post {

   //declaring variables
   private static ArrayList<Comment> comments = new ArrayList<>();
   private String comment;
   private int commentID;
   private static int lastCommentID = Endorsement.getLastEndorsementID();
   private List <Comment> commentsCommentsList = new ArrayList<>();

   /**
    * a constructor for objects in class Comment.
    * @param handle
    * @param id
    * @param message
    */
   public Comment(String handle, int id, String message) {
      super(handle, id);
      this.comment = message;
      this.commentID = ++lastCommentID;
   }

   //getter for the static list of comments
   public static ArrayList<Comment> getCommentsList() {
      return comments;
   }

   //getter for comment
   public String getComment() {
```

```java
      return comment;
   }

   //getter for commentID
   public int getCommentID() {
      return commentID;
   }

   //getter for the list of replies of each comment.
   public List<Comment> getCommentsCommentsList() {
      return commentsCommentsList;
   }

   /**
    * This method adds a comment to the static list of comments.
    */
   public static void addComment(Comment comment) {
      comments.add(comment);
   }

   /**
    * This method displays the Comment.
    */
   public String toString() {
      return "@Comment " + getHandle() + ": " + comment;
   }


}
```

3 Endorsement.java

```java
package socialmedia;

import java.util.ArrayList;

/**
 * This class stores information on each Endorsement in the social media platform.
 * Endorsements automatically replicate the endorsed message. The endorsed message cannot be endorsed again.
 * This class is a subclass of Post class.
 * @author Aisha
 * @author Saida
 *
 */
@SuppressWarnings("serial")
public class Endorsement extends Post {

   //declaring variables
   private String endorsedMessage;
   private int endorsementID;
   private static int lastEndorsementID = Post.getLastID();
   private static ArrayList<Endorsement> endorsements = new ArrayList<>();

   /**
```

```java
     * constructor for objects of class Endorsement.
     * @param handle
     * @param id
     */
    public Endorsement(String handle, int id) {
       super(handle, id);
       this.endorsementID = ++lastEndorsementID;
       this.endorsedMessage = getMessage();
    }

    //setter for endorsedMessage
    public void setEndorsedMessage(String endorsedMessage) {
       this.endorsedMessage = endorsedMessage;
    }

    //getter for the static list of endorsements
    public static ArrayList<Endorsement> getEndorsementsList() {
       return endorsements;
    }

    //getter for endorsementID
    public int getEndorsementID() {
       return endorsementID;
    }

    //getter for lastEndorsementID()
    public static int getLastEndorsementID() {
       return lastEndorsementID;
    }

     /**
      * This method adds an endorsement to the static list of endorsements.
      */
     public static void addEndorsement(Endorsement endorsement){
         endorsements.add(endorsement);
     }

    /**
     * This method displays the Endorsement.
     */
    public String toString() {
       return "EP@ " + getHandle() + ": " + endorsedMessage;
    }

}
```

4 Post.java

```java
package socialmedia;

import java.util.ArrayList;
import java.util.List;

/**
```

```java
 * This class stores information on each post of an individual Account in the social media platforms.
 * Posts can be created, removed and have a caption up to 100 characters.
 * The class is a subclass of Account class.
 * @author Aisha
 * @author Saida
 *
 */
@SuppressWarnings("serial")
public class Post extends Account {

   //declaring variables
    private static int lastID = 0;
   private String message;
   private int id;
    private static ArrayList<Post> posts = new ArrayList<>();
    private List<Endorsement> endorsementsList = new ArrayList<>();
    private List<Comment> commentsList = new ArrayList<>();

   /**
    * constructor for objects of class Post.
    * @param handle
    * @param message
    */
   public Post(String handle, String message) {
      super(handle);
      this.id = ++lastID;
      this.message = message;
   }

   /**
    * 2nd constructor for objects of class Post.
    * @param handle
    * @param id
    */
   public Post(String handle, int id) {
      super(handle);
      this.id = id;
   }

   //setter for message
   public void setMessage(String postCaption) {
      this.message = postCaption;
   }

   //getter for message
   public String getMessage() {
      return message;
   }

   //getter for postID
   public int getPostID() {
      return id;
   }

   //getter for the list of endorsements of each post
```

```
    public List<Endorsement> getEachPostEndorsementList() {
      return endorsementsList;
    }

    //getter for the list of comments of each post
    public List<Comment> getEachCommentList() {
      return commentsList;
    }

     //getter for the static list of posts
     public static ArrayList<Post> getPostsList() {
       return posts;
     }

     /**
      * This method adds a post into the static posts list.
      */
     public static void addPost(Post post){
         posts.add(post);
     }

     /**
      * This method gets the lastID of Post.
      * @return lastID
      */
     public static int getLastID() {
       return lastID;
     }

     /**
      * This method adds an endorsement to the endorsement list of every post object created.
      */
     public void addEndorsementtoList(Endorsement endorsement) {
         endorsementsList.add(endorsement);
     }

     /**
      * This method returns the number of endorsements for each post object created.
      * @return number of endorsements
      */
     public int getNumberOfEndorsements() {
         return endorsementsList.size();
     }

     /**
      * This method adds a comment to the comment list of every post object created.
      */
     public void addCommentToList(Comment comment) {
         commentsList.add(comment);
     }

     /**
      * This method returns the number of comments for each post object created.
      * @return number of comments
      */
```

```java
    public int getNumberOfComments() {
       return commentsList.size();
    }

    /**
     * This method displays the Post.
     * @return toString of Post
     */
    public String toString() {
       return "Post[name = " + getHandle() + ", message = " +
       message + ", ID = " + id + "]";
   }




}
```

5 SocialMedia.java

```java
package socialmedia;

import java.util.ArrayList;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;

/**
 * SocialMedia is a compiling implementor of
 * the SocialMediaPlatform interface.
 * @author Aisha
 * @author Saida
 * @version 1.0
 */
@SuppressWarnings("serial")
public class SocialMedia implements SocialMediaPlatform {

    public int createAccount(String handle, String description) throws IllegalHandleException,
        InvalidHandleException {

      if (handle.isEmpty() || handle.length() > 30 || handle.contains(" ")) {
         throw new InvalidHandleException("Sorry, the handle you provided is empty, too long or contains
             whitespace.");
      }

      for (int i = 0; i < Account.getAccountsList().size(); i++) {
         Account account = Account.getAccountsList().get(i);
            if (account.getHandle().equals(handle)) {
               throw new IllegalHandleException("Sorry, the handle already exists.");
         }
      }

      Account newAccount = new Account(handle, description);
      Account.addAccount(newAccount);
      return newAccount.getID();

}


   @Override
   public void removeAccount(String handle) throws HandleNotRecognisedException {

      boolean checkHandle = false;

      for (int i = 0; i < Account.getAccountsList().size(); i++) {
         Account account = Account.getAccountsList().get(i);
         if (account.getHandle().equals(handle)) {
            Account.getAccountsList().remove(account);
            Endorsement.getEndorsementsList().removeAll(account.getAccountEndorsementsList());
            checkHandle = true;
         }
      }

      for (int i = 0; i < Post.getPostsList().size(); i++) {
         Post post = Post.getPostsList().get(i);
         if (post.getHandle().equals(handle)) {
            Post.getPostsList().remove(post);
            post.getEachPostEndorsementList().clear();
            checkHandle = true;
         }
      }

       if (!checkHandle) {
         throw new HandleNotRecognisedException("Sorry, the handle does not exist.");
      }

   }

   @Override
   public void updateAccountDescription(String handle, String description) throws
       HandleNotRecognisedException {

      boolean checkHandle = false;

       for (int i = 0; i < Account.getAccountsList().size(); i++) {
           Account account = Account.getAccountsList().get(i);
           if (account.getHandle().equals(handle)) {
               account.setDescription(description);
               checkHandle = true;
           }

       }   if (!checkHandle) {
            throw new HandleNotRecognisedException("Sorry, the handle does not exist.");
       }

   }

```

```

    @Override
    public int getNumberOfAccounts() {
       return Account.getAccountsList().size();

    }

    @Override
    public int getTotalOriginalPosts() {
       return Post.getPostsList().size();

    }

    @Override
    public int getTotalEndorsmentPosts() {
       return Endorsement.getEndorsementsList().size();
    }

    @Override
    public int getTotalCommentPosts() {
       return Comment.getCommentsList().size();
    }

    @Override
    public int createAccount(String handle) throws IllegalHandleException, InvalidHandleException {

       if (handle.isEmpty() || handle.length() > 30 || handle.contains(" ")) {
          throw new InvalidHandleException("Sorry, the handle you provided is empty, too long or contains
              whitespace.");
       }

       for (int i = 0; i < Account.getAccountsList().size(); i++) {
          Account account = Account.getAccountsList().get(i);
             if (account.getHandle().equals(handle)) {
                throw new IllegalHandleException("Sorry, the handle already exists.");
          }
       }

         Account newAccount = new Account(handle);
         Account.addAccount(newAccount);
         return newAccount.getID();

    }

    @Override
    public void removeAccount(int id) throws AccountIDNotRecognisedException {

       boolean checkID = false;

       for (int i = 0; i < Account.getAccountsList().size(); i++) {
          Account account = Account.getAccountsList().get(i);
             if (account.getID() == id) {
                checkID = true;
                Account.getAccountsList().remove(account);
                Post.getPostsList().removeAll(account.getEachPostList());
```

```java
                Endorsement.getEndorsementsList().removeAll(account.getAccountEndorsementsList());
            }
        }

      if (!checkID) {
            throw new AccountIDNotRecognisedException("Sorry, the ID is not recognised.");
      }

    }

    @Override
    public void changeAccountHandle(String oldHandle, String newHandle) throws HandleNotRecognisedException,
        IllegalHandleException, InvalidHandleException {

      boolean checkHandle = false;

      for (int i = 0; i < Account.getAccountsList().size(); i++) {
        Account account = Account.getAccountsList().get(i);
        if (account.getHandle().equals(newHandle)) {
          throw new IllegalHandleException("Sorry, this handle already exists.");
          } else if(oldHandle.isEmpty() || oldHandle.length() > 30 || oldHandle.contains(" ")) {
            throw new InvalidHandleException("Sorry, the handle you provided is empty, too long or
                contains whitespace.");
          } else if(account.getHandle().equals(oldHandle)) {
            account.setHandle(newHandle);
            checkHandle = true;
          }

      }  if(!checkHandle) {
            throw new HandleNotRecognisedException("Sorry, this handle does not exist.");
       }    else if(newHandle.isEmpty() || newHandle.length() > 30 || newHandle.contains(" ")) {
            throw new InvalidHandleException("Sorry, the handle you provided is empty, too long or contains
                whitespace.");
       }
    }

    @Override
    public String showAccount(String handle) throws HandleNotRecognisedException {

      boolean checkHandle = false;

       for (int i = 0; i < Account.getAccountsList().size();i++) {
         Account account = Account.getAccountsList().get(i);
         if (account.getHandle().equals(handle)) {
          String formattedString = String.format("ID: " + account.getID() + "\nAccount: " +
              account.getHandle() + "\nDescription: "
          + account.getDescription() + "\nPost Count: " + account.getEachPostList().size() +
              "\nEndorsement Count: " + account.getAccountEndorsementsList().size());
          checkHandle = true;
          return formattedString;
         }
       }

        if (!checkHandle) {
         throw new HandleNotRecognisedException("Sorry, the handle does not exist.");
    }

   return null;

}

@Override
public int createPost(String handle, String message) throws HandleNotRecognisedException,
    InvalidPostException {

   boolean checkHandle = false;

   if (message.isEmpty() || message.length() > 100) {
      throw new InvalidPostException("Sorry, your post is either empty or too long.");
   }

   for (int i = 0; i < Account.getAccountsList().size(); i++) {
       Account account = Account.getAccountsList().get(i);
       if (account.getHandle().equals(handle)) {
          Post newPost = new Post(handle,message);
          Post.addPost(newPost);
          account.getEachPostList().add(newPost);
          checkHandle = true;
          return newPost.getPostID();
       }
   }

      if (!checkHandle) {
         throw new HandleNotRecognisedException("Sorry, the handle does not exist.");
   }

   return 0;

}

@Override
public int endorsePost(String handle, int id) throws HandleNotRecognisedException,
    PostIDNotRecognisedException, NotActionablePostException {

   boolean checkHandle = false;
   boolean checkID = false;

   for (int i = 0; i < Endorsement.getEndorsementsList().size(); i++) {
      Endorsement endorsement = Endorsement.getEndorsementsList().get(i);
      if (endorsement.getEndorsementID() == id) {
         throw new NotActionablePostException("You cannot endorse an endorsement.");
      }
   }

   for (int i = 0; i < Post.getPostsList().size(); i++) {
      Post post = Post.getPostsList().get(i);
      if (post.getPostID() == id) {
      checkID = true;
      }
         for (Comment comment: post.getEachCommentList()) {
          if (comment.getCommentID() == id) {
            checkID = true;
          }
        }
      }

      if(!checkID) {
        throw new PostIDNotRecognisedException("Sorry, the ID is not recognised.");
      }

      for (int i = 0; i < Account.getAccountsList().size(); i++) {
        Account account = Account.getAccountsList().get(i);
        if (account.getHandle().equals(handle)) {
          checkHandle = true;
        }
      }

      if(!checkHandle) {
        throw new HandleNotRecognisedException("Sorry, the handle does not exist.");
      }

      for (int i = 0; i < Post.getPostsList().size(); i++) {
        for (Account account: Account.getAccountsList()) {
          Post post = Post.getPostsList().get(i);
          if (post.getPostID() == id && account.getHandle().equals(post.getHandle())) {
            Endorsement newEndorsement = new Endorsement(handle, id);
            Endorsement.addEndorsement(newEndorsement);
            post.addEndorsementtoList(newEndorsement);
            account.addAccountEndorsement(newEndorsement);
            return newEndorsement.getEndorsementID();
          }
        }
      }

      return 0;

    }

    @Override
    public int commentPost(String handle, int id, String message) throws HandleNotRecognisedException,
        PostIDNotRecognisedException, NotActionablePostException, InvalidPostException {

      boolean checkHandle = false;
      boolean checkID = false;

      for (int i = 0; i < Endorsement.getEndorsementsList().size(); i++) {
        Endorsement endorsement = Endorsement.getEndorsementsList().get(i);
        if (endorsement.getEndorsementID() == id) {
          throw new NotActionablePostException("You cannot endorse an endorsement.");
          }
        }

      @SuppressWarnings("unused")
      boolean checkComment = false;
        for (int j = 0; j < Post.getPostsList().size(); j++) {
          Post post = Post.getPostsList().get(j);
          if (post.getPostID() == id) {
             checkID = true;
       } else {
          for (Comment comment: post.getEachCommentList()) {
             if (comment.getCommentID() == id) {
                checkID = true;
                checkComment = true;
             }
          }
       }
    }

     if(!checkID){
        throw new PostIDNotRecognisedException("Sorry, the ID is not recognised.");
     }

     for (int i = 0; i < Post.getPostsList().size();i++) {
        Post post = Post.getPostsList().get(i);
        if (post.getHandle().equals(handle)) {
          checkHandle = true;
        }
     }

    if(!checkHandle){
       throw new HandleNotRecognisedException("Sorry, the handle does not exist.");
    } else if (message.isEmpty() || message.length() > 100) {
       throw new InvalidPostException("Sorry, your post is either empty or too long.");
    }

    for (int i = 0; i < Post.getPostsList().size();i++) {
        Post post = Post.getPostsList().get(i);
        if (post.getPostID() == id) {
          Comment comment = new Comment(handle, id, message);
          Comment.addComment(comment);
          post.addCommentToList(comment);
          comment.getCommentsCommentsList().add(comment);
          return comment.getCommentID();
         }
    }

    return 0;
 }

 @Override
 public void deletePost(int id) throws PostIDNotRecognisedException {

    boolean checkPostID = false;

       for (int i = 0; i < Post.getPostsList().size(); i++) {
             Post post = Post.getPostsList().get(i);
             if (post.getPostID() == id) {
             checkPostID = true;
             Post.getPostsList().remove(post);
             Endorsement.getEndorsementsList().removeAll(post.getEachPostEndorsementList());
```

```java
               }
            }

         if(!checkPostID){
            throw new PostIDNotRecognisedException("Sorry, the ID is not recognised.");
         }

   }

   @Override
   public String showIndividualPost(int id) throws PostIDNotRecognisedException {

      boolean checkPostID = false;

      for (int i = 0; i < Post.getPostsList().size();i++) {
            Post post = Post.getPostsList().get(i);
            if (post.getPostID() == id) {
               String formattedString = String.format("ID: " + post.getPostID() + "\nAccount: "+
                    post.getHandle()
               + "\nNo.of Endorsements: " + post.getNumberOfEndorsements() +
               " || No.of Comments: " + post.getNumberOfComments() +"\n" + post.getMessage());
               checkPostID = true;
               return formattedString;
            }
      }

      if (!checkPostID){
         throw new PostIDNotRecognisedException("Sorry, the ID is not recognised.");
      }

      return null;

   }

   @Override
   public StringBuilder showPostChildrenDetails(int id)
         throws PostIDNotRecognisedException, NotActionablePostException {

      boolean checkID = false;

      for (int i = 0; i < Endorsement.getEndorsementsList().size(); i++) {
         Endorsement endorsement = Endorsement.getEndorsementsList().get(i);
         if (endorsement.getEndorsementID() == id) {
            throw new NotActionablePostException("You cannot endorse an endorsement.");
            }
         }

      StringBuilder postChildren = new StringBuilder();

         for (int i = 0; i < Post.getPostsList().size(); i++) {
         Post post = Post.getPostsList().get(i);
         if (post.getPostID() == id) {
            checkID = true;
            postChildren.append(showIndividualPost(id));
            for (int j = 0; j < post.getEachCommentList().size(); j++) {
```

```
            Comment comment = post.getEachCommentList().get(j);
            postChildren.append("\n| > ID: ").append(comment.getCommentID()).append("\n");
            postChildren.append("| Account: ").append(comment.getHandle()).append("\n");
            postChildren.append("| No.endorsements: ").append(post.getNumberOfEndorsements()).append(" |
                No. comments: ").append(post.getNumberOfComments());
            postChildren.append("\n").append(comment.getComment());
            for (int k = 0; k < comment.getCommentsCommentsList().size();k++) {
              Comment commentOfcomment = comment.getCommentsCommentsList().get(k);
              postChildren.append("\n | > ID: ").append(commentOfcomment.getCommentID()).append("\n");
              postChildren.append(" |  Account: ").append(commentOfcomment.getHandle()).append("\n");
              postChildren.append(" |  No.endorsements:
                  ").append(commentOfcomment.getNumberOfEndorsements()).append(" | No. comments:
                  ").append(commentOfcomment.getNumberOfComments());
              postChildren.append("\n").append(commentOfcomment.getComment());
          }
        }
      }
    }

      if(!checkID){
        throw new PostIDNotRecognisedException("Sorry, the ID is not recognised.");
      }

    return postChildren;
  }

  @Override
  public int getMostEndorsedPost() {

    int max = 0;
    Post mostEndorsedPost = null;

    for (int i = 0; i < Post.getPostsList().size(); i++) {
      Post post = Post.getPostsList().get(i);
      int endorsementsSize = post.getEachPostEndorsementList().size();
      if (endorsementsSize > max) {
        max = endorsementsSize;
        mostEndorsedPost = post;
      }

    }

    System.out.println("The most endorsed post is: " + mostEndorsedPost);

    return mostEndorsedPost.getPostID();
  }

  @Override
  public int getMostEndorsedAccount() {

    int max = 0;
    Account mostEndorsedAccount = null;

    for (int i = 0; i < Account.getAccountsList().size(); i++) {
      Account account = Account.getAccountsList().get(i);
```

```
            int endorsementsSize = account.getAccountEndorsementsList().size();
            if (endorsementsSize > max) {
               max = endorsementsSize;
               mostEndorsedAccount = account;
            }

         }

         System.out.println("The most endorsed account is: " + mostEndorsedAccount);


         return mostEndorsedAccount.getID();
      }

      @Override
      public void erasePlatform() {
         Account.getAccountsList().clear();
         Post.getPostsList().clear();
         Endorsement.getEndorsementsList().clear();
         Comment.getCommentsList().clear();


      }

      @Override
      public void savePlatform(String filename) throws IOException {

         try {

         FileOutputStream fos = new FileOutputStream(filename);
         ObjectOutputStream OOS = new ObjectOutputStream(fos);

         ArrayList <Account> accounts = Account.getAccountsList();
         ArrayList<Post> posts = Post.getPostsList();
         ArrayList<Endorsement> endorsements = Endorsement.getEndorsementsList();
         ArrayList<Comment> comments = Comment.getCommentsList();

         OOS.writeObject(accounts);

         OOS.writeObject(posts);

         OOS.writeObject(endorsements);

         OOS.writeObject(comments);

         OOS.close();
         fos.close();

         System.out.println("Platform saved.");

         } catch (IOException e) {
            e.printStackTrace();
         }
       }

```

```java
    @Override
    @SuppressWarnings("unchecked")
    public void loadPlatform(String filename) throws IOException, ClassNotFoundException {

       try {

     FileInputStream fis = new FileInputStream(filename);

     ObjectInputStream OIS = new ObjectInputStream(fis);

     ArrayList<Account> accounts = (ArrayList<Account>) OIS.readObject();

     ArrayList<Post> posts = (ArrayList<Post>) OIS.readObject();

     ArrayList<Endorsement> endorsements = (ArrayList<Endorsement>) OIS.readObject();

     ArrayList<Comment> comments = (ArrayList<Comment>) OIS.readObject();




     for (Post post : posts) {
         System.out.println(post);
     }
     for (Account account : accounts) {
         System.out.println(account);
     }
     for (Comment comment : comments) {
         System.out.println(comment);
     }
     for (Endorsement endorsement : endorsements) {
         System.out.println(endorsement);
     }
      OIS.close();

      fis.close();

 } catch (IOException | ClassNotFoundException e) {

     e.printStackTrace();

    }


    }


 }
```