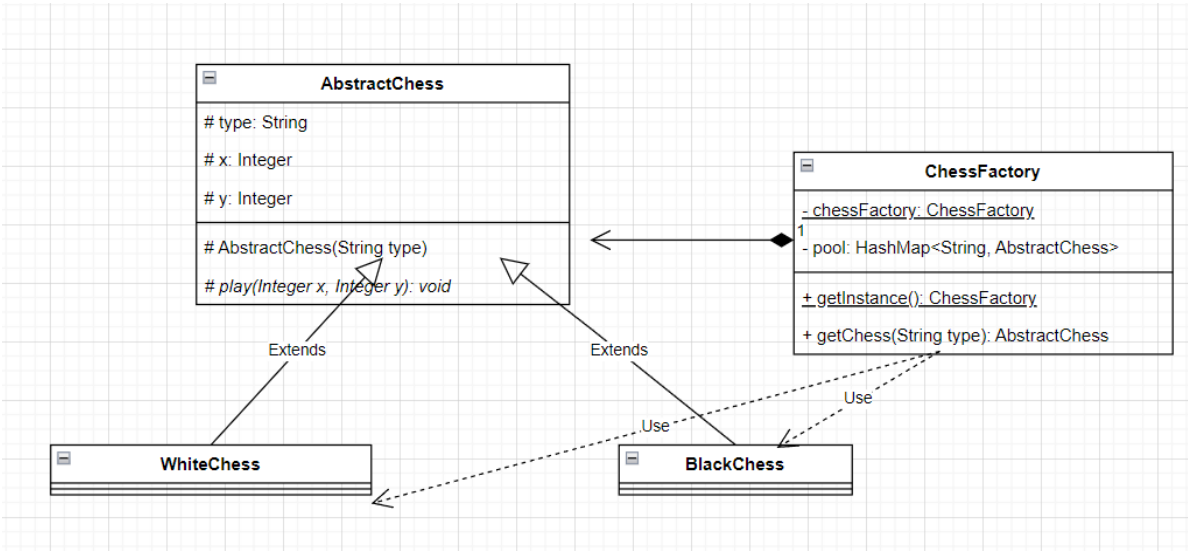
9-16周设计模式

flyweight pattern

UML



code

```
package seu.assignment.flyweight;

/**
 * @ClassName: AbstractChess
 * @Description: java类描述
 * @Author: 11609
 * @Date: 2022/11/3 16:19:55
 * @Input:
 * @Output:
 */
abstract class AbstractChess {
   protected String type;
   protected Integer x;
   protected Integer y;
   protected AbstractChess(String type) {
      this.type = type;
   }

   protected abstract void play(Integer x, Integer y);
}
```

```
package seu.assignment.flyweight;

/**
 * @ClassName: BlackChess
 * @Description: java类描述
 * @Author: 11609
 * @Date: 2022/11/3 16:31:02
```

```
 * @Input:
 * @Output:
 */
class BlackChess extends AbstractChess {

   protected BlackChess() {
      super("Black");
      System.out.println("---------Black Chess Constructed");
   }

   @Override
   protected void play(Integer x, Integer y) {
      this.x = x;
      this.y = y;
      System.out.println("---------Black Chess on: " + "(" + this.x + ", " +
this.y + ")");
   }
}
```

```
package seu.assignment.flyweight;

import java.util.HashMap;

/**
 * @ClassName: ChessFactory
 * @Description: java类描述
 * @Author: 11609
 * @Date: 2022/11/3 16:32:47
 * @Input:
 * @Output:
 */
class ChessFactory {
   private static ChessFactory chessFactory = new ChessFactory();
   private final HashMap<String, AbstractChess> pool = new HashMap<>();
   public static ChessFactory getInstance() {
      return chessFactory;
   }
   public AbstractChess getChess(String type) {
      AbstractChess chess = this.pool.get(type);
      if (chess == null) {
         if (type.equals("White")) {
            chess = new WhiteChess();
         } else if (type.equals("Black")) {
            chess = new BlackChess();
         } else {
            chess = null;
         }
      }
      if (chess != null) {
         this.pool.put(type, chess);
      }
      return chess;
   }
}
```

```java
package seu.assignment.flyweight;

/**
 * @ClassName: WhiteChess
 * @Description: java类描述
 * @Author: 11609
 * @Date: 2022/11/3 16:26:32
 * @Input:
 * @Output:
 */
class WhiteChess extends AbstractChess {

   protected WhiteChess() {
      super("White");
      System.out.println("---------White Chess Constructed");
   }

   @Override
   protected void play(Integer x, Integer y) {
      this.x = x;
      this.y = y;
      System.out.println("---------White Chess on: " + "(" + this.x + ", " +
this.y + ")");
   }
}
```

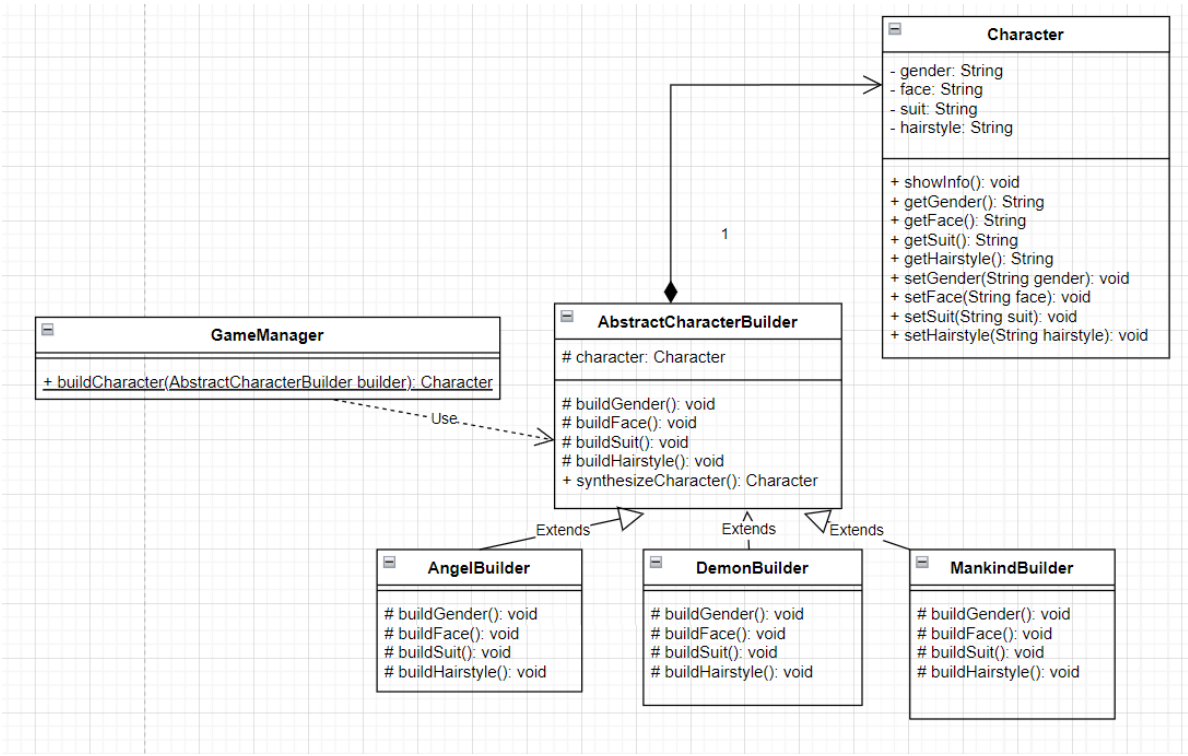
```java
package seu.assignment.flyweight;

/**
 * @ClassName: Client
 * @Description: java类描述
 * @Author: 11609
 * @Date: 2022/11/3 16:39:02
 * @Input:
 * @Output:
 */
class Client {
   public static void main(String[] args) {
      ChessFactory chessFactory = ChessFactory.getInstance();
      AbstractChess white1 = chessFactory.getChess("White");
      AbstractChess white2 = chessFactory.getChess("White");
      AbstractChess white3 = chessFactory.getChess("White");
      AbstractChess black1 = chessFactory.getChess("Black");
      AbstractChess black2 = chessFactory.getChess("Black");
      white1.play(1, 7);
      white2.play(2, 5);
      white3.play(3, 7);
      black1.play(2, 0);
      black2.play(9, 0);
   }
}
```

output

```
---------White Chess Constructed
---------Black Chess Constructed
---------White Chess on: (1, 7)
---------White Chess on: (2, 5)
---------White Chess on: (3, 7)
---------Black Chess on: (2, 0)
---------Black Chess on: (9, 0)
```

builder pattern

UML



code

```java
package seu.assignment.builder;

/**
 * @ClassName: AbstractCharacterBuilder
 * @Description: java类描述
 * @Author: 11609
 * @Date: 2022/11/3 21:51:23
 * @Input:
 * @Output:
 */
abstract class AbstractCharacterBuilder {
    protected Character character = new Character();
    protected abstract void buildGender();
    protected abstract void buildFace();
    protected abstract void buildSuit();
```

```
    protected abstract void buildHairstyle();

    public Character synthesizeCharacter() {
        buildGender();
        buildFace();
        buildSuit();
        buildHairstyle();
        return character;
    };
}
```

```
package seu.assignment.builder;

/**
 * @ClassName: AngelBuilder
 * @Description: java类描述
 * @Author: 11609
 * @Date: 2022/11/3 21:51:44
 * @Input:
 * @Output:
 */
class AngelBuilder extends AbstractCharacterBuilder {
    @Override
    protected void buildGender() {
        this.character.setGender("Female(Angel)");
    }

    @Override
    protected void buildFace() {
        this.character.setFace("Cute(Angel)");
    }

    @Override
    protected void buildSuit() {
        this.character.setSuit("Goddess(Angel)");
    }

    @Override
    protected void buildHairstyle() {
        this.character.setHairstyle("WavyHair(Angel)");
    }
}
```

```
package seu.assignment.builder;

/**
 * @ClassName: Character
 * @Description: java类描述
 * @Author: 11609
 * @Date: 2022/11/3 21:51:29
 * @Input:
 * @Output:
 */
class Character {
```

```
    private String gender;
    private String face;
    private String suit;
    private String hairstyle;

    public void showInfo() {
        System.out.println("------------Character info: ---------------");
        System.out.println("------------Gender: " + this.gender);
        System.out.println("------------Face: " + this.face);
        System.out.println("------------Suit: " + this.suit);
        System.out.println("------------Hairstyle: " + this.hairstyle);
        System.out.println("------------End info: ---------------");
        System.out.println();
    }

    public String getGender() {
        return gender;
    }

    public void setGender(String gender) {
        this.gender = gender;
    }

    public String getFace() {
        return face;
    }

    public void setFace(String face) {
        this.face = face;
    }

    public String getSuit() {
        return suit;
    }

    public void setSuit(String suit) {
        this.suit = suit;
    }

    public String getHairstyle() {
        return hairstyle;
    }

    public void setHairstyle(String hairstyle) {
        this.hairstyle = hairstyle;
    }
}
```

```
package seu.assignment.builder;

/**
 * @ClassName: DemonBuilder
 * @Description: java类描述
 * @Author: 11609
 * @Date: 2022/11/3 21:51:53
```

```
 * @Input:
 * @Output:
 */
class DemonBuilder extends AbstractCharacterBuilder {
    @Override
    protected void buildGender() {
        this.character.setGender("Male(Demon)");
    }

    @Override
    protected void buildFace() {
        this.character.setFace("Furious(Demon)");
    }

    @Override
    protected void buildSuit() {
        this.character.setSuit("Dominator(Demon)");
    }

    @Override
    protected void buildHairstyle() {
        this.character.setHairstyle("ShortHair(Demon)");
    }
}
```

```
package seu.assignment.builder;

/**
 * @ClassName: GameManager
 * @Description: java类描述
 * @Author: 11609
 * @Date: 2022/11/3 21:51:37
 * @Input:
 * @Output:
 */
class GameManager {
    public static Character buildCharacter(AbstractCharacterBuilder builder) {
        return builder.synthesizeCharacter();
    }

    public static void main(String[] args) {
        Character angel = GameManager.buildCharacter(new AngelBuilder());
        Character demon = GameManager.buildCharacter(new DemonBuilder());
        Character mankind = GameManager.buildCharacter(new MankindBuilder());
        angel.showInfo();
        demon.showInfo();
        mankind.showInfo();
    }
}
```

```
package seu.assignment.builder;

/**
 * @ClassName: MankindBuilder
```

```
 * @Description: java类描述
 * @Author: 11609
 * @Date: 2022/11/3 21:52:01
 * @Input:
 * @Output:
 */
class MankindBuilder extends AbstractCharacterBuilder {
    @Override
    protected void buildGender() {
        this.character.setGender("Female(Mankind)");
    }

    @Override
    protected void buildFace() {
        this.character.setFace("Expressionless(Mankind)");
    }

    @Override
    protected void buildSuit() {
        this.character.setSuit("Monarch(Mankind)");
    }

    @Override
    protected void buildHairstyle() {
        this.character.setHairstyle("CurlyHair(Mankind)");
    }
}
```

output

```
------------Character info: ---------------
------------Gender: Female(Angel)
------------Face: Cute(Angel)
------------Suit: Goddess(Angel)
------------Hairstyle: WavyHair(Angel)
------------End info: ---------------

------------Character info: ---------------
------------Gender: Male(Demon)
------------Face: Furious(Demon)
------------Suit: Dominator(Demon)
------------Hairstyle: ShortHair(Demon)
------------End info: ---------------

------------Character info: ---------------
------------Gender: Female(Mankind)
------------Face: Expressionless(Mankind)
------------Suit: Monarch(Mankind)
------------Hairstyle: CurlyHair(Mankind)
------------End info: ---------------
```

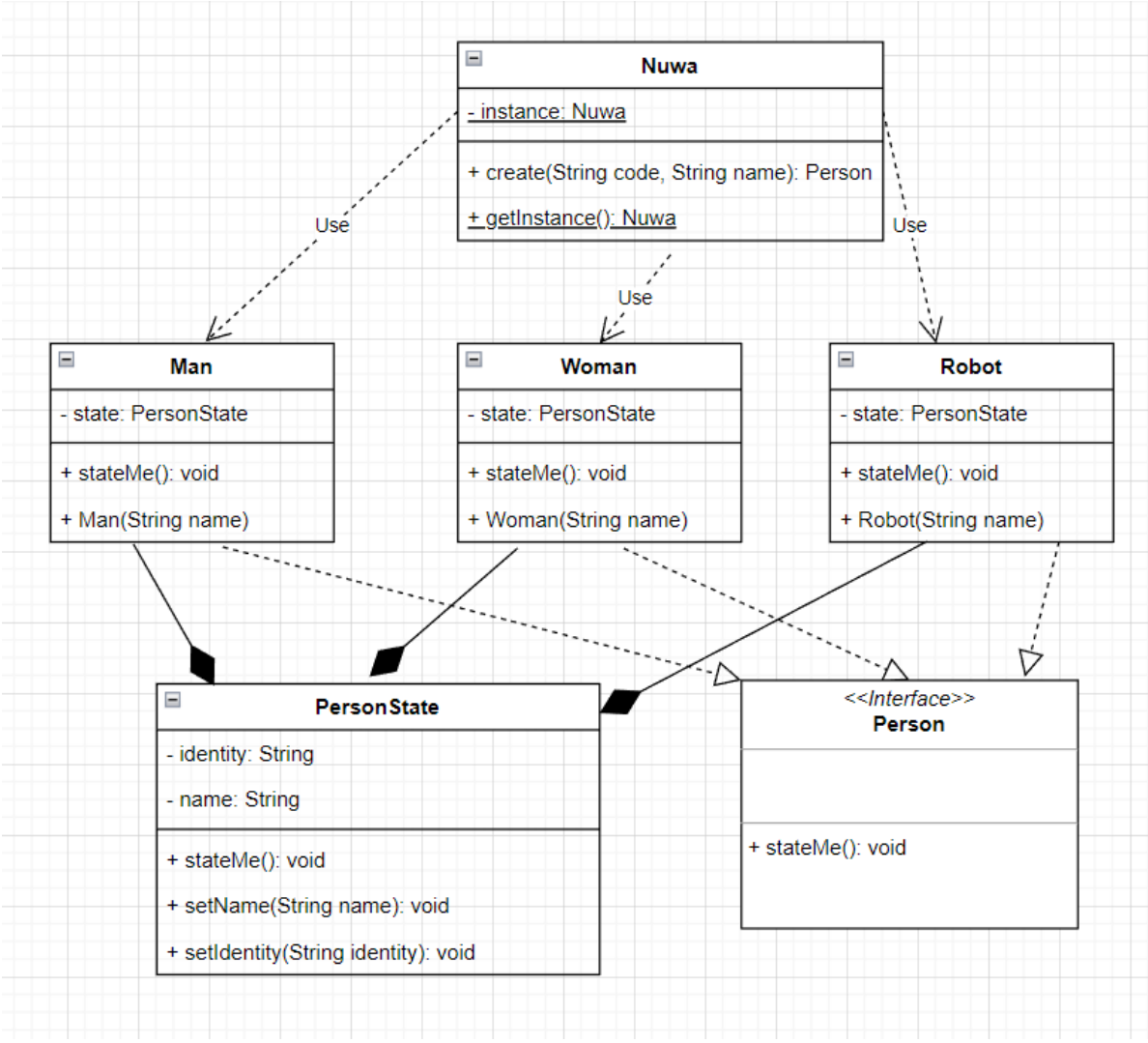
simple factory pattern

extra

在每次添加一个可被创造的Person具体类时，都需要对于Nuwa类中的创建函数进行修改——添加相应类的if else判断。

```
public Person create(String code, String name) {
    Person person = null;
    if (Objects.equals(code, "M")) {
        System.out.println("---------------Create Man: " + name);
        person = new Man(name);
    } else if (Objects.equals(code, "W")) {
        System.out.println("---------------Create Woman: " + name);
        person = new Woman(name);
    } else if (Objects.equals(code, "R")) {
        System.out.println("---------------Create Robot: " + name);
        person = new Robot(name);
    } else {
        System.out.println("----- Error Create!!! -----");
    }
    return person;
}
```

UML



code

```java
package seu.assignment.simple_factory;

/**
 * @ClassName: Man
 * @Description: java类描述
 * @Author: 11609
 * @Date: 2022/11/4 21:08:38
 * @Input:
 * @Output:
 */
class Man implements Person {

    private final PersonState state = new PersonState();

    public Man(String name) {
        state.setName(name);
        state.setIdentity("Man");
    }

    @Override
    public void stateMe() {
        state.stateMe();
    }
}
```

```java
package seu.assignment.simple_factory;

/**
 * @ClassName: Woman
 * @Description: java类描述
 * @Author: 11609
 * @Date: 2022/11/4 21:08:43
 * @Input:
 * @Output:
 */
class Woman implements Person {

    private final PersonState state = new PersonState();

    public Woman(String name) {
        state.setName(name);
        state.setIdentity("Woman");
    }

    @Override
    public void stateMe() {
        state.stateMe();
    }
}
```

```java
package seu.assignment.simple_factory;
```

```java
/**
 * @ClassName: Robot
 * @Description: java类描述
 * @Author: 11609
 * @Date: 2022/11/4 21:08:50
 * @Input:
 * @Output:
 */
class Robot implements Person {

    private final PersonState state = new PersonState();

    public Robot(String name) {
        state.setName(name);
        state.setIdentity("Robot");
    }

    @Override
    public void stateMe() {
        state.stateMe();
    }
}
```

```java
package seu.assignment.simple_factory;

public interface Person {
    void stateMe();
}
```

```java
package seu.assignment.simple_factory;

/**
 * @ClassName: PersonState
 * @Description: java类描述
 * @Author: 11609
 * @Date: 2022/11/4 21:08:59
 * @Input:
 * @Output:
 */
class PersonState {

   private String identity;
   private String name;

   public void stateMe() {
      System.out.println("-------------State: ----------");
      System.out.println("-------------Identity: " + this.identity);
      System.out.println("-------------Name: " + this.name);
      System.out.println("-------------State End");
      System.out.println();
   }

   public PersonState() {}
```

```
    public void setIdentity(String identity) {
        this.identity = identity;
    }

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

```
package seu.assignment.simple_factory;

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

/**
 * @ClassName: Nuwa
 * @Description: java类描述
 * @Author: 11609
 * @Date: 2022/11/4 21:08:32
 * @Input:
 * @Output:
 */
class Nuwa {
    private static final Nuwa instance = new Nuwa();

    public Person create(String code, String name) {
        Person person = null;
        if (Objects.equals(code, "M")) {
            System.out.println("---------------Create Man: " + name);
            person = new Man(name);
        } else if (Objects.equals(code, "W")) {
            System.out.println("---------------Create Woman: " + name);
            person = new Woman(name);
        } else if (Objects.equals(code, "R")) {
            System.out.println("---------------Create Robot: " + name);
            person = new Robot(name);
        } else {
            System.out.println("----- Error Create!!! -----");
        }
        return person;
    }

    public static Nuwa getInstance() {
        return instance;
    }

    public static void main(String[] args) {
        Nuwa nuwa = Nuwa.getInstance();
        List<Person> people = new ArrayList<>();
        people.add(nuwa.create("M", "William"));
        people.add(nuwa.create("W", "Cayena"));
        people.add(nuwa.create("W", "Camelia"));
        people.add(nuwa.create("R", "Atori"));
```

```
        people.forEach(Person::stateMe);
    }
}
```

output

```
---------------Create Man: William
---------------Create Woman: Cayena
---------------Create Woman: Camelia
---------------Create Robot: Atori
-------------State: ----------
-------------Identity: Man
-------------Name: William
-------------State End

-------------State: ----------
-------------Identity: Woman
-------------Name: Cayena
-------------State End

-------------State: ----------
-------------Identity: Woman
-------------Name: Camelia
-------------State End

-------------State: ----------
-------------Identity: Robot
-------------Name: Atori
-------------State End
```

TRIPLEDES

extra

工厂模式创建

使用上有一点状态模式

```
KeyGenerator kg = KeyGenerator.getInstance(KEY_ALGORITHM);
      // 初始化密钥生成器
      kg.init(168);
```

```
Cipher cipher = Cipher.getInstance(CIPHER_ALGORITHM);
      // 初始化，设置为加密模式
      cipher.init(Cipher.ENCRYPT_MODE, k);
```

code

```
package seu.assignment.des_factory;

import java.io.BufferedOutputStream;
import java.io.FileOutputStream;
import java.security.Key;
import javax.crypto.Cipher;
import javax.crypto.KeyGenerator;
import javax.crypto.SecretKey;
import javax.crypto.SecretKeyFactory;
import javax.crypto.spec.DESedeKeySpec;
import com.sun.org.apache.xml.internal.security.utils.Base64;

/**
 * @ClassName: Test
 * @Description: java类描述
 * @Author: 11609
 * @Date: 2022/11/4 21:39:23
 * @Input:
 * @Output:
 */

public class DESedeCoder {
    /**
     * 密钥算法
     * */
    public static final String KEY_ALGORITHM = "DESede";

    /**
     * 加密/解密算法/工作模式/填充方式
     * */
    public static final String CIPHER_ALGORITHM = "DESede/ECB/PKCS5Padding";

    /**
     *
     * 生成密钥
     *
     * @return byte[] 二进制密钥
     * */
    public static byte[] initkey() throws Exception {

        // 实例化密钥生成器
        KeyGenerator kg = KeyGenerator.getInstance(KEY_ALGORITHM);
        // 初始化密钥生成器
        kg.init(168);
        // 生成密钥
        SecretKey secretKey = kg.generateKey();
        // 获取二进制密钥编码形式

        byte[] key = secretKey.getEncoded();
        BufferedOutputStream keystream =
                new BufferedOutputStream(new FileOutputStream("DESedeKey.dat"));
        keystream.write(key, 0, key.length);
        keystream.flush();
```

```
        keystream.close();

        return key;
    }

    /**
     * 转换密钥
     *
     * @param key
     *            二进制密钥
     * @return Key 密钥
     * */
    public static Key toKey(byte[] key) throws Exception {
        // 实例化Des密钥
        DESedeKeySpec dks = new DESedeKeySpec(key);
        // 实例化密钥工厂
        SecretKeyFactory keyFactory = SecretKeyFactory
                .getInstance(KEY_ALGORITHM);
        // 生成密钥
        SecretKey secretKey = keyFactory.generateSecret(dks);
        return secretKey;
    }

    /**
     * 加密数据
     *
     * @param data
     *            待加密数据
     * @param key
     *            密钥
     * @return byte[] 加密后的数据
     * */
    public static byte[] encrypt(byte[] data, byte[] key) throws Exception {
        // 还原密钥
        Key k = toKey(key);
        // 实例化
        Cipher cipher = Cipher.getInstance(CIPHER_ALGORITHM);
        // 初始化，设置为加密模式
        cipher.init(Cipher.ENCRYPT_MODE, k);
        // 执行操作
        return cipher.doFinal(data);
    }

    /**
     * 解密数据
     *
     * @param data
     *            待解密数据
     * @param key
     *            密钥
     * @return byte[] 解密后的数据
     * */
    public static byte[] decrypt(byte[] data, byte[] key) throws Exception {
        // 欢迎密钥
        Key k = toKey(key);
```

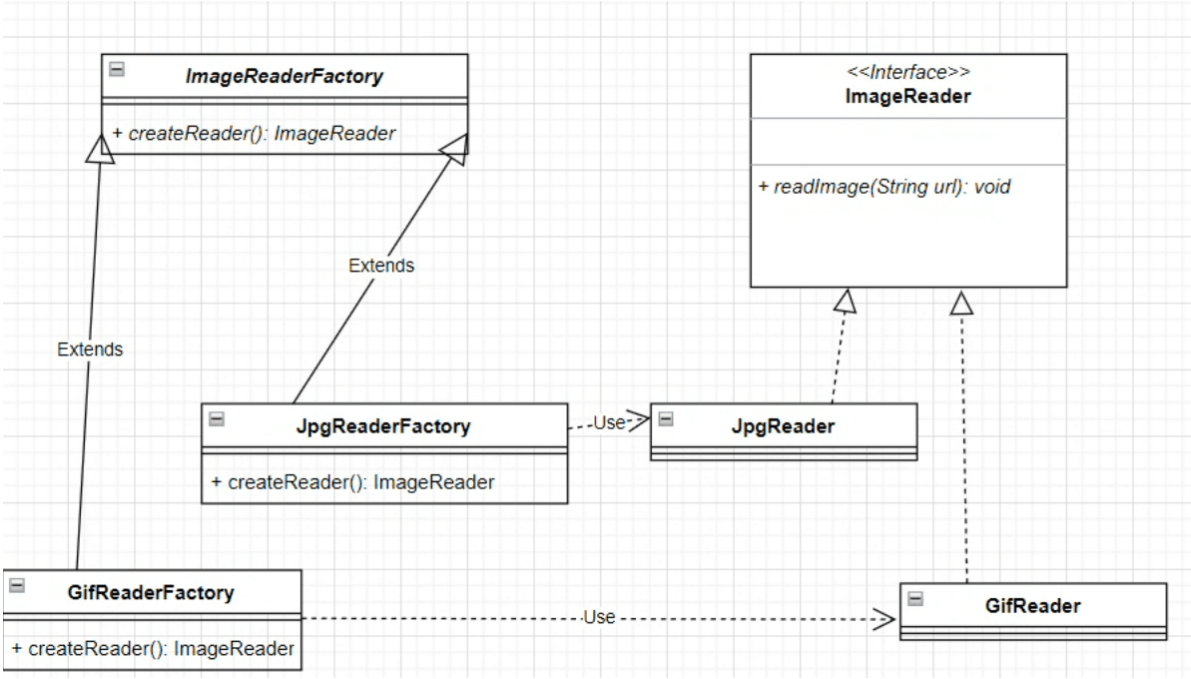
```
        // 实例化
        Cipher cipher = Cipher.getInstance(CIPHER_ALGORITHM);
        // 初始化，设置为解密模式
        cipher.init(Cipher.DECRYPT_MODE, k);
        // 执行操作
        return cipher.doFinal(data);
    }

    /**
     * 进行加解密的测试
     *
     * @throws Exception
     */
    public static void main(String[] args) throws Exception {
        String str = "Hello, design pattern.";
        System.out.println("原文: " + str);
        // 初始化密钥
        byte[] key = DESedeCoder.initkey();
        System.out.println("密钥: " + Base64.encode(key));
        // 加密数据
        byte[] data = DESedeCoder.encrypt(str.getBytes(), key);
        System.out.println("加密后: " + Base64.encode(data));
        // 解密数据
        data = DESedeCoder.decrypt(data, key);
        System.out.println("解密后: " + new String(data));
    }
}
```

output

```
原文: Hello, design pattern.
密钥: O9xX7zQ0fOB/DV2P6b+AKff3wjQBtTSi
加密后: zCoY1gCv0jvCb6pEmfoXpP/uB2oGOhh3
解密后: Hello, design pattern.
```

factory method pattern

UML



code

```java
package seu.assignment.factory_method;

public interface ImageReader {
    void readImage(String url);
}
```

```java
package seu.assignment.factory_method;



/**
 * @ClassName: ImageReaderFactory
 * @Description: java类描述
 * @Author: 11609
 * @Date: 2022/11/9 10:58:58
 * @Input:
 * @Output:
 */
abstract class ImageReaderFactory {
   public abstract ImageReader createImageReader();
}
```

```java
package seu.assignment.factory_method;

/**
 * @ClassName: JpgReader
 * @Description: java类描述
 * @Author: 11609
 * @Date: 2022/11/9 11:01:30
 * @Input:
 * @Output:
 */
```

```
class JpgReader implements ImageReader {
   @Override
   public void readImage(String url) {
      if (url.endsWith(".jpg")) {
         System.out.println("---------------JPG Reader Working On: " + url);
         return;
      }
      System.out.println("---------------Error Format For JPG Reader!");
   }
}
```

```
package seu.assignment.factory_method;

/**
 * @ClassName: GifReader
 * @Description: java类描述
 * @Author: 11609
 * @Date: 2022/11/9 11:04:21
 * @Input:
 * @Output:
 */
class GifReader implements ImageReader {
   @Override
   public void readImage(String url) {
      if (url.endsWith(".gif")) {
         System.out.println("---------------GIF Reader Working On: " + url);
         return;
      }
      System.out.println("---------------Error Format For GIF Reader!");
   }
}
```

```
package seu.assignment.factory_method;

/**
 * @ClassName: GifReaderFactory
 * @Description: java类描述
 * @Author: 11609
 * @Date: 2022/11/9 11:06:40
 * @Input:
 * @Output:
 */
class GifReaderFactory extends ImageReaderFactory {
   @Override
   public ImageReader createImageReader() {
      return new GifReader();
   }
}
```

```
package seu.assignment.factory_method;

/**
 * @ClassName: JpgReaderFactory
```

```
 * @Description: java类描述
 * @Author: 11609
 * @Date: 2022/11/9 11:04:59
 * @Input:
 * @Output:
 */
class JpgReaderFactory extends ImageReaderFactory {
   @Override
   public ImageReader createImageReader() {
      return new JpgReader();
   }
}
```

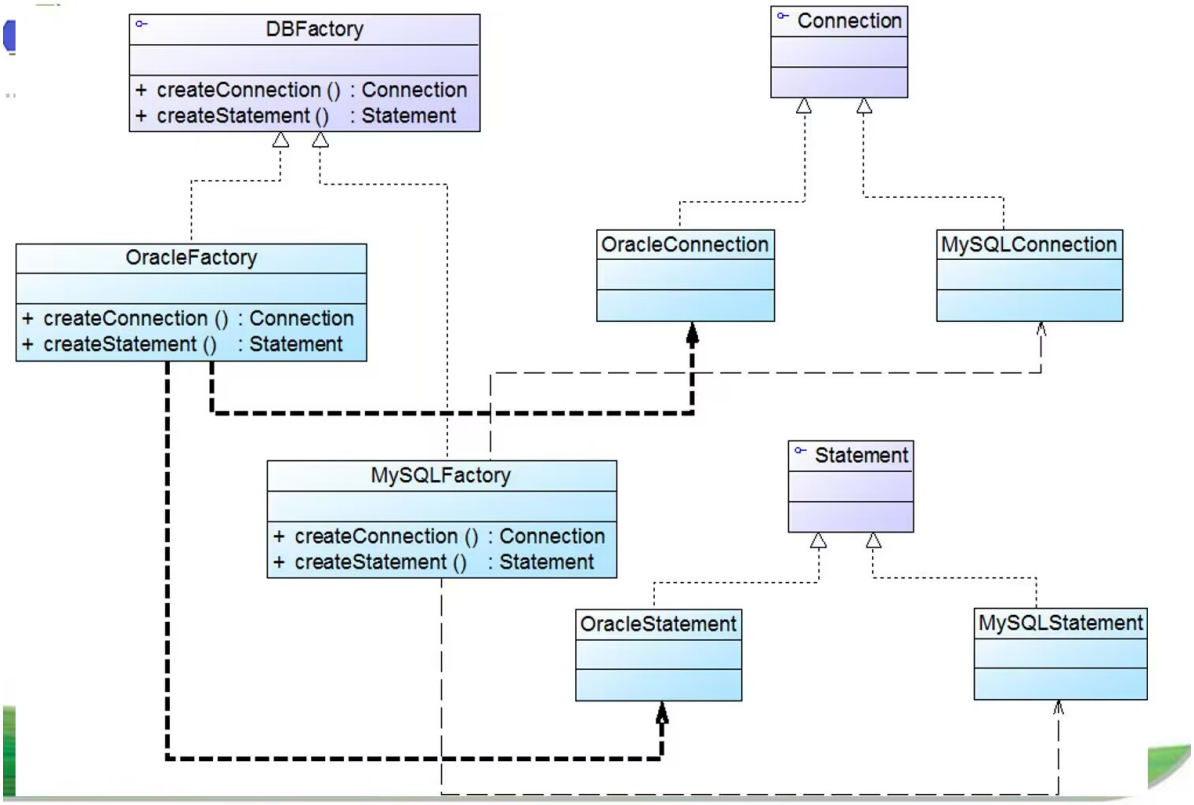
```
package seu.assignment.factory_method;

/**
 * @ClassName: Client
 * @Description: java类描述
 * @Author: 11609
 * @Date: 2022/11/9 11:10:40
 * @Input:
 * @Output:
 */
class Client {
   public static void main(String[] args) {
      String gifFile = "camelia.gif";
      String jpgFile = "cayena.jpg";

      ImageReaderFactory gifFactory = new GifReaderFactory();
      ImageReaderFactory jpgFactory = new JpgReaderFactory();

      ImageReader gifReader = gifFactory.createImageReader();
      ImageReader jpgReader = jpgFactory.createImageReader();

      // ----- for gif reader
      gifReader.readImage(gifFile);
      gifReader.readImage(jpgFile);

      // ----- for jpg reader
      jpgReader.readImage(gifFile);
      jpgReader.readImage(jpgFile);
   }
}
```

output

```
---------------GIF Reader Working On: camelia.gif
---------------Error Format For GIF Reader!
---------------Error Format For JPG Reader!
---------------JPG Reader Working On: cayena.jpg
```

DATABASE

UML



code

```
package seu.assignment.abstract_factory;

public interface Statement {
   void execute();
}
```

```
package seu.assignment.abstract_factory;

public interface DBFactory {
    Connection createConnection();
    Statement createStatement();
}
```

```
package seu.assignment.abstract_factory;

public interface Connection {
    void connect();
}
```

```
package seu.assignment.abstract_factory;

/**
 * @ClassName: MySQLFactory
 * @Description: java类描述
```

```
 * @Author: 11609
 * @Date: 2022/11/10 16:32:44
 * @Input:
 * @Output:
 */
class MySQLFactory implements DBFactory {
   @Override
   public Connection createConnection() {
      System.out.println("-------------MySQLConnection Created!");
      return new MySQLConnection();
   }

   @Override
   public Statement createStatement() {
      System.out.println("-------------MySQLStatement Created!");
      return new MySQLStatement();
   }
}
```

```
package seu.assignment.abstract_factory;

/**
 * @ClassName: MySQLConnection
 * @Description: java类描述
 * @Author: 11609
 * @Date: 2022/11/10 16:33:12
 * @Input:
 * @Output:
 */
class MySQLConnection implements Connection {
   @Override
   public void connect() {
      System.out.println("-------------MySQL Connected!");
   }
}
```

```
package seu.assignment.abstract_factory;

/**
 * @ClassName: MySQLStatement
 * @Description: java类描述
 * @Author: 11609
 * @Date: 2022/11/10 16:33:36
 * @Input:
 * @Output:
 */
class MySQLStatement implements Statement {
    @Override
    public void execute() {
        System.out.println("-------------MySQL Execution!");
    }
}
```

```java
package seu.assignment.abstract_factory;

/**
 * @ClassName: OracleFactory
 * @Description: java类描述
 * @Author: 11609
 * @Date: 2022/11/10 16:32:06
 * @Input:
 * @Output:
 */
class OracleFactory implements DBFactory {
    @Override
    public Connection createConnection() {
        System.out.println("-------------OracleConnection Created!");
        return new OracleConnection();
    }

    @Override
    public Statement createStatement() {
        System.out.println("-------------OracleStatement Created!");
        return new OracleStatement();
    }
}
```

```java
package seu.assignment.abstract_factory;

/**
 * @ClassName: OracleConnection
 * @Description: java类描述
 * @Author: 11609
 * @Date: 2022/11/10 16:33:02
 * @Input:
 * @Output:
 */
class OracleConnection implements Connection {
    @Override
    public void connect() {
        System.out.println("-------------Oracle Connected!");
    }
}
```

```java
package seu.assignment.abstract_factory;

/**
 * @ClassName: OracleStatement
 * @Description: java类描述
 * @Author: 11609
 * @Date: 2022/11/10 16:33:24
 * @Input:
 * @Output:
 */
class OracleStatement implements Statement {
    @Override
    public void execute() {
```

```
        System.out.println("-------------Oracle Executed!");
    }
}
```

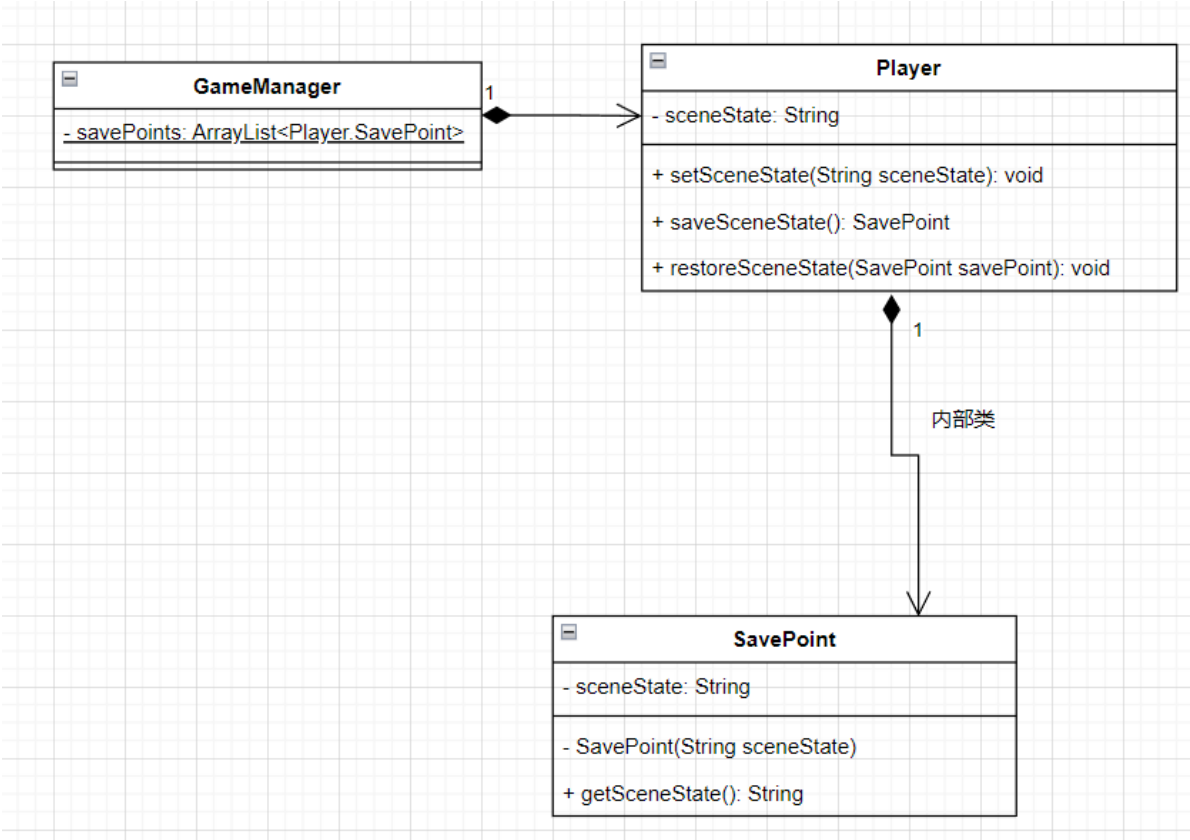
```
package seu.assignment.abstract_factory;

import com.sun.org.apache.xpath.internal.operations.Or;

/**
 * @ClassName: Client
 * @Description: java类描述
 * @Author: 11609
 * @Date: 2022/11/10 16:39:29
 * @Input:
 * @Output:
 */
class Client {
   public static void main(String[] args) {
      DBFactory mySQLFactory = new MySQLFactory();
      DBFactory oracleFactory = new OracleFactory();

      mySQLFactory.createConnection().connect();
      mySQLFactory.createStatement().execute();

      oracleFactory.createConnection().connect();
      oracleFactory.createStatement().execute();
   }
}
```

output

```
-------------MySQLConnection Created!
-------------MySQL Connected!
-------------MySQLStatement Created!
-------------MySQL Execution!
-------------OracleConnection Created!
-------------Oracle Connected!
-------------OracleStatement Created!
-------------Oracle Executed!
```

memento pattern

UML



code

```
package seu.assignment.memento;

import java.util.ArrayList;

/**
 * @ClassName: GameManager
 * @Description: java类描述
 * @Author: 11609
 * @Date: 2022/11/4 15:17:59
 * @Input:
 * @Output:
 */
class GameManager {
   private static ArrayList<Player.SavePoint> savePoint = new ArrayList<>();

   public static void main(String[] args) {
      Player player = new Player();
      player.setSceneState("Level1");
      player.setSceneState("Level2");
      savePoint.add(player.saveSceneState());
      player.setSceneState("Level3");
      player.restoreSceneState(savePoint.get(0));
      player.setSceneState("Level3");
      player.setSceneState("Level4");
      savePoint.add(player.saveSceneState());
      player.setSceneState("Level5");
      player.restoreSceneState(savePoint.get(1));

   }
}
```

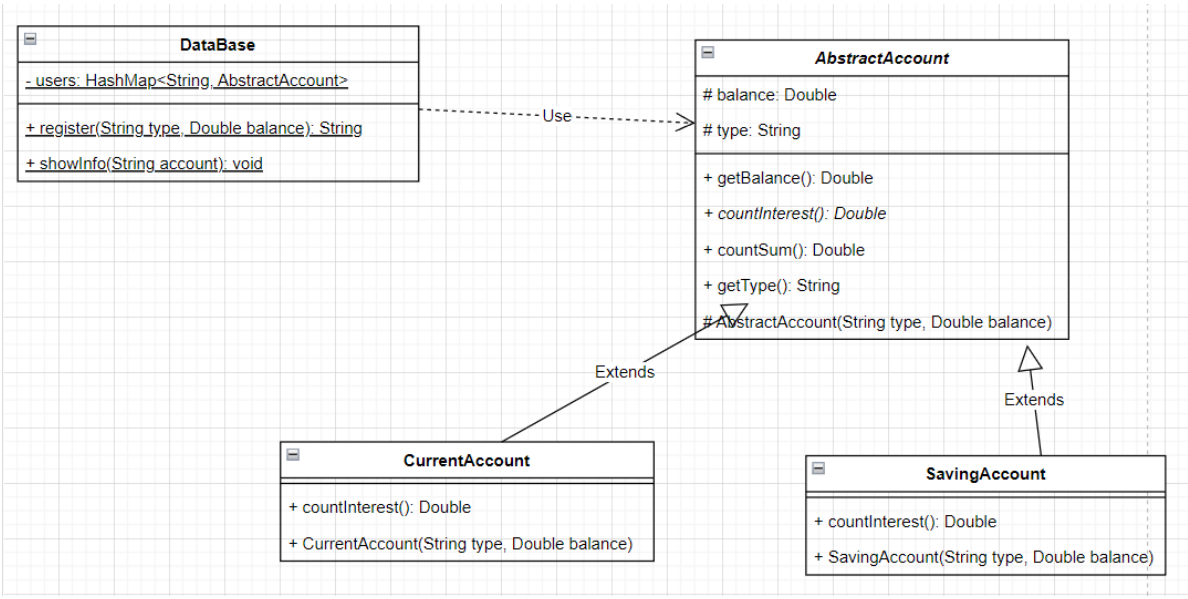
```java
package seu.assignment.memento;

/**
 * @ClassName: Player
 * @Description: java类描述
 * @Author: 11609
 * @Date: 2022/11/4 14:52:46
 * @Input:
 * @Output:
 */
class Player {
   private String sceneState;

   public void setSceneState(String sceneState) {
      System.out.println("------------Player: Scene State Altered From: " +
this.sceneState + " to " + sceneState + " -----------");
      this.sceneState = sceneState;
   }
   public void restoreSceneState(SavePoint savePoint) {
      System.out.println("------------Player: Scene State Restored From: " +
this.sceneState + " to " + savePoint.getSceneState() + " -----------");
      this.sceneState = savePoint.getSceneState();
   }
   public SavePoint saveSceneState() {
      System.out.println("------------Player: Scene State Saved!-----------");
      return new SavePoint(this.sceneState);
   }
   class SavePoint {
      private final String sceneState;

      private SavePoint(String sceneState) {
         this.sceneState = sceneState;
      }

      public String getSceneState() {
         return sceneState;
      }

   }
}
```

output

```
------------Player: Scene State Altered From: null to Level1 -----------
------------Player: Scene State Altered From: Level1 to Level2 -----------
------------Player: Scene State Saved!-----------
------------Player: Scene State Altered From: Level2 to Level3 -----------
------------Player: Scene State Restored From: Level3 to Level2 -----------
------------Player: Scene State Altered From: Level2 to Level3 -----------
------------Player: Scene State Altered From: Level3 to Level4 -----------
------------Player: Scene State Saved!-----------
------------Player: Scene State Altered From: Level4 to Level5 -----------
------------Player: Scene State Restored From: Level5 to Level4 -----------
```

template method pattern

UML



code

```java
package seu.assignment.template;

/**
 * @ClassName: AbstractAccount
 * @Description: java类描述
 * @Author: 11609
 * @Date: 2022/11/4 16:55:20
 * @Input:
 * @Output:
 */
abstract class AbstractAccount {
    protected Double balance;
    protected String type;

    protected AbstractAccount(String type, Double balance) {
        this.type = type;
        this.balance = balance;
    }

    public Double getBalance() {
        return balance;
    }

    public String getType() {
        return type;
    }

    public Double countSum() {
        System.out.println("----------------Type: " + this.type);
```

```
        return balance + countInterest();
    }

    public abstract Double countInterest();
}
```

```
package seu.assignment.template;

/**
 * @ClassName: CurrentAccount
 * @Description: java类描述
 * @Author: 11609
 * @Date: 2022/11/4 17:05:49
 * @Input:
 * @Output:
 */
class CurrentAccount extends AbstractAccount {

    public CurrentAccount(String type, Double balance) {
        super(type, balance);
    }

    @Override
    public Double countInterest() {
        System.out.println("----------------Current Account Interest: balance *
5");
        return this.balance * 5;
    }
}
```

```
package seu.assignment.template;

/**
 * @ClassName: SavingAccount
 * @Description: java类描述
 * @Author: 11609
 * @Date: 2022/11/4 17:00:06
 * @Input:
 * @Output:
 */
class SavingAccount extends AbstractAccount {

    public SavingAccount(String type, Double balance) {
        super(type, balance);
    }

    @Override
    public Double countInterest() {
        System.out.println("----------------Saving Account Interest: balance *
10");
        return this.balance * 10;
    }
}
```

```java
package seu.assignment.template;

import java.util.HashMap;
import java.util.Random;

/**
 * @ClassName: DataBase
 * @Description: java类描述
 * @Author: 11609
 * @Date: 2022/11/4 17:15:18
 * @Input:
 * @Output:
 */
class DataBase {
    private static final HashMap<String, AbstractAccount> users = new HashMap<>
();
    public static String register(String type, Double balance) {
        String account = "";
        Random random = new Random();
        for (int i = 0; i < 9; i++) {
            account += random.nextInt(10);
        }
        if (type == "SavingAccount") {
            System.out.println("----------------Create SavingAccount");
            users.put(account, new SavingAccount(type, balance));
        } else if (type == "CurrentAccount") {
            System.out.println("----------------Create CurrentAccount");
            users.put(account, new CurrentAccount(type, balance));
        }
        else {
            System.out.println("-----------------Error registering");
            return null;
        }

        return account;
    }
    public static void showInfo(String account) {
        AbstractAccount user = users.get(account);
        if (user == null) {
            System.out.println("----------------No such user");
            return;
        }
        System.out.println("----------------User Info: ----------------");
        System.out.println("----------------Type: " + user.getType());
        System.out.println("----------------Type: " + user.getBalance());
        System.out.println("----------------Type: " + user.countSum());
        System.out.println("----------------End Info: ----------------");
        System.out.println();
    }

    public static void main(String[] args) {
        final String savingAccount = "SavingAccount";
        final String currentAccount = "CurrentAccount";
        String accountSaving = DataBase.register(savingAccount, 1000.0);
        String accountCurrent = DataBase.register(currentAccount, 1000.0);
```

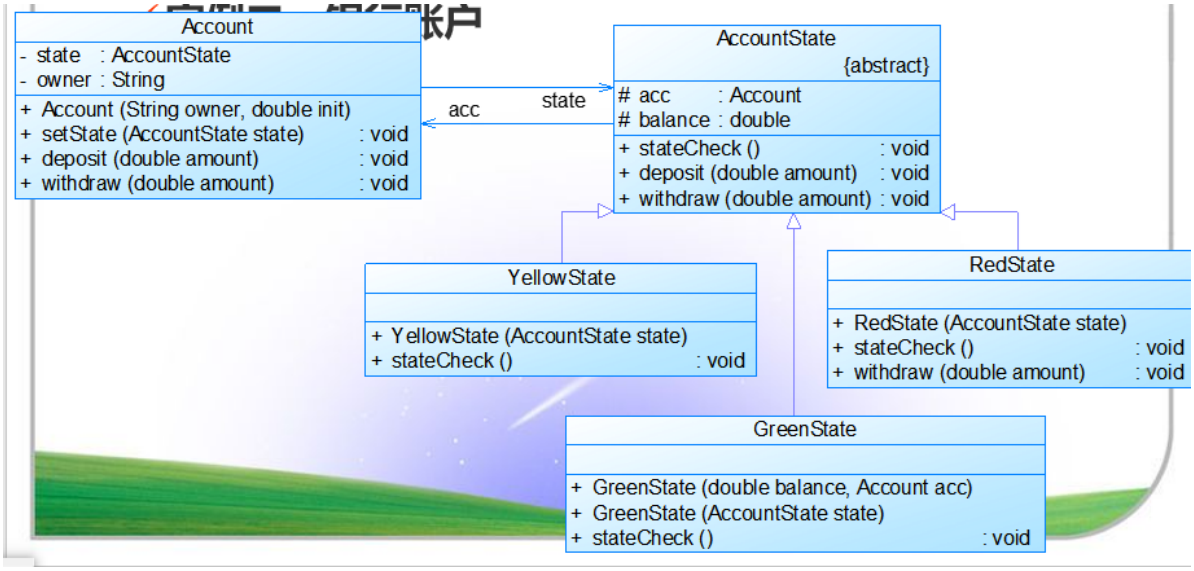
```
        DataBase.showInfo(accountSaving);
        DataBase.showInfo(accountCurrent);
    }
}
```

output

```
----------------Create SavingAccount
----------------Create CurrentAccount
----------------User Info: ----------------
----------------Type: SavingAccount
----------------Type: 1000.0
----------------Type: SavingAccount
----------------Saving Account Interest: balance * 10
----------------Type: 11000.0
----------------End Info: ----------------

----------------User Info: ----------------
----------------Type: CurrentAccount
----------------Type: 1000.0
----------------Type: CurrentAccount
----------------Current Account Interest: balance * 5
----------------Type: 6000.0
----------------End Info: ----------------
```

Bank

UML



code

```
package seu.assignment.state2;

/**
 * @ClassName: Account
 * @Description: java类描述
 * @Author: 11609
 * @Date: 2022/11/8 16:09:10
 * @Input:
```

```
 * @Output:
 */
class Account {
   private AccountState state;
   private String owner;

   public Account(String owner, double init) {
      this.owner = owner;
      this.state = new GreenState(init, this);
   }

   public void setState(AccountState state) {
      this.state = state;
   }

   public void deposit(double amount) {
      System.out.println("-------------Deposit: " + amount);

      state.deposit(amount);
   }

   public void withdraw(double amount) {
      System.out.println("-------------Withdraw: " + amount);

      state.withdraw(amount);
   }

}
```

```
package seu.assignment.state2;

/**
 * @ClassName: AccountState
 * @Description: java类描述
 * @Author: 11609
 * @Date: 2022/11/8 16:08:35
 * @Input:
 * @Output:
 */
abstract class AccountState {
   protected Account acc;
   protected double balance;

   public abstract void stateCheck();

   public void deposit(double amount) {
      this.balance += amount;
      stateCheck();
   }

   public void withdraw(double amount) {
      this.balance -= amount;
      stateCheck();
```

```
    }

}
```

```
package seu.assignment.state2;

/**
 * @ClassName: GreenState
 * @Description: java类描述
 * @Author: 11609
 * @Date: 2022/11/8 16:54:37
 * @Input:
 * @Output:
 */
class GreenState extends AccountState {
    public GreenState(double balance, Account acc) {
        this.balance = balance;
        this.acc = acc;
    }

    public GreenState(AccountState state) {
        this.balance = state.balance;
        this.acc = state.acc;
    }

    @Override
    public void stateCheck() {
        if (balance < -1000) {
            System.out.println("-------------Green To Red State");
            acc.setState(new RedState(this));
        }
        if (balance < 0 && balance >= -1000) {
            System.out.println("-------------Green To Yellow State");
            acc.setState(new YellowState(this));
        }
    }
}
```

```
package seu.assignment.state2;

/**
 * @ClassName: RedState
 * @Description: java类描述
 * @Author: 11609
 * @Date: 2022/11/8 16:59:15
 * @Input:
 * @Output:
 */
class RedState extends AccountState {
    public RedState(AccountState state) {
        this.balance = state.balance;
        this.acc = state.acc;
    }
```

```
    @Override
    public void stateCheck() {
        if (balance >= 0) {
            System.out.println("-------------Red To Green State");
            acc.setState(new GreenState(this));
        }
        if (balance < 0 && balance >= -1000) {
            System.out.println("-------------Red To Yellow State");
            acc.setState(new YellowState(this));
        }
    }

    @Override
    public void withdraw(double amount) {
        System.out.println("-------------Fail to withdraw : Red State");
    }
}
```

```
package seu.assignment.state2;

/**
 * @ClassName: YellowState
 * @Description: java类描述
 * @Author: 11609
 * @Date: 2022/11/8 16:23:34
 * @Input:
 * @Output:
 */
class YellowState extends AccountState {

    public YellowState(AccountState state) {
        this.acc = state.acc;
        this.balance = state.balance;
    }

    @Override
    public void stateCheck() {
        if (balance < -1000) {
            System.out.println("-------------Yellow To Red State");
            acc.setState(new RedState(this));
        }
        if (balance >= 0) {
            System.out.println("-------------Yellow To Green State");
            acc.setState(new GreenState(this));
        }
    }
}
```

```
package seu.assignment.state2;

/**
 * @ClassName: Client
 * @Description: java类描述
 * @Author: 11609
```

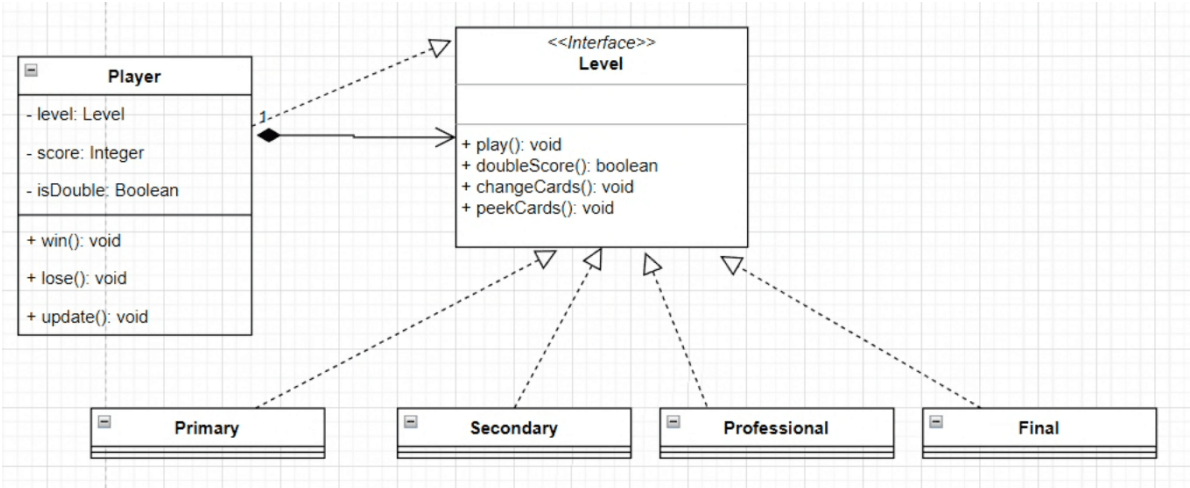
```
 * @Date: 2022/11/10 20:39:37
 * @Input:
 * @Output:
 */
class Client {
    public static void main(String[] args) {
        Account camelia = new Account("Camelia", 1200);
        // ---- Green
        camelia.deposit(200);
        camelia.withdraw(1500);
        // ---- Yellow | Green
        camelia.deposit(100);
        camelia.withdraw(3000);
        // ---- Red
        camelia.deposit(100);
        camelia.withdraw(3000);
    }
}
```

output

```
-------------Deposit: 200.0
-------------Withdraw: 1500.0
-------------Green To Yellow State
-------------Deposit: 100.0
-------------Yellow To Green State
-------------Withdraw: 3000.0
-------------Green To Red State
-------------Deposit: 100.0
-------------Withdraw: 3000.0
-------------Fail to withdraw : Red State
```

state pattern

UML



code

```
package seu.assignment.state;

public interface Level {
   void play();
   boolean doubleScore();
   void changeCards();
   void peekCards();
}
```

```
package seu.assignment.state;

/**
 * @ClassName: Player
 * @Description: java类描述
 * @Author: 11609
 * @Date: 2022/11/8 15:33:52
 * @Input:
 * @Output:
 */
class Player implements Level {

   private Level level = new Primary();

   private Integer score = 0;

   private Boolean isDouble = false;

   public void update() {
      if (score == 0) {
         level = new Primary();
      } else if (score == 1) {
         level = new Secondary();
      } else if (score == 2) {
         level = new Professional();
      } else if (score == 3) {
         level = new Final();
      } else {
         level = new Final();
      }
   }

   public void win() {
      System.out.println("---------------Game Win");
      if (isDouble) {
         score += 2;
         isDouble = false;
         update();
         return;
      }
      score += 1;
      update();
   }
```

```
    public void lose() {
        System.out.println("---------------Game Lose");
        isDouble = false;
        score = score > 0 ? score - 1 : 0;
        update();
    }

    @Override
    public void play() {
        level.play();
    }

    @Override
    public boolean doubleScore() {
        if (level.doubleScore()) {
            isDouble = true;
        }
        return isDouble;
    }

    @Override
    public void changeCards() {
        level.changeCards();
    }

    @Override
    public void peekCards() {
        level.peekCards();
    }

    public static void main(String[] args) {
        Player player = new Player();

        // ------- for primary -------

        player.play();
        player.doubleScore();
        player.changeCards();
        player.peekCards();

        player.win();

        // ------- for secondary -------


        player.play();
        player.doubleScore();
        player.changeCards();
        player.peekCards();

        player.win();

        // ------- for final -------

        player.play();
```

```
        player.doubleScore();
        player.changeCards();
        player.peekCards();

        player.lose();

        // ------- for professional -------

        player.play();
        player.doubleScore();
        player.changeCards();
        player.peekCards();

        player.lose();
    }
}
```

```
package seu.assignment.state;

/**
 * @ClassName: Primary
 * @Description: java类描述
 * @Author: 11609
 * @Date: 2022/11/8 15:34:17
 * @Input:
 * @Output:
 */
class Primary implements Level {
    @Override
    public void play() {
        System.out.println("---------------Play Game");
    }

    @Override
    public boolean doubleScore() {
        System.out.println("---------------!!!!Not Allowed");
        return false;
    }

    @Override
    public void changeCards() {
        System.out.println("---------------!!!!Not Allowed");
    }

    @Override
    public void peekCards() {
        System.out.println("---------------!!!!Not Allowed");
    }
}
```

```
package seu.assignment.state;

/**
 * @ClassName: Secondary
```

```
 * @Description: java类描述
 * @Author: 11609
 * @Date: 2022/11/8 15:34:23
 * @Input:
 * @Output:
 */
class Secondary implements Level {
   @Override
   public void play() {
      System.out.println("---------------Play Game");
   }

   @Override
   public boolean doubleScore() {
      System.out.println("---------------Double Scores");
      return true;
   }

   @Override
   public void changeCards() {
      System.out.println("---------------!!!!Not Allowed");
   }

   @Override
   public void peekCards() {
      System.out.println("---------------!!!!Not Allowed");
   }
}
```

```
package seu.assignment.state;

/**
 * @ClassName: Professional
 * @Description: java类描述
 * @Author: 11609
 * @Date: 2022/11/8 15:34:33
 * @Input:
 * @Output:
 */
class Professional implements Level {
    @Override
    public void play() {
        System.out.println("---------------Play Game");
    }

    @Override
    public boolean doubleScore() {
        System.out.println("---------------Double Scores");
        return true;
    }

    @Override
    public void changeCards() {
        System.out.println("---------------Change Cards");
    }
```

```
    @Override
    public void peekCards() {
        System.out.println("---------------!!!!Not Allowed");
    }
}
```

```
package seu.assignment.state;

/**
 * @ClassName: Final
 * @Description: java类描述
 * @Author: 11609
 * @Date: 2022/11/8 15:34:40
 * @Input:
 * @Output:
 */
class Final implements Level {
    @Override
    public void play() {
        System.out.println("---------------Play Game");
    }

    @Override
    public boolean doubleScore() {
        System.out.println("---------------Double Scores");
        return true;
    }

    @Override
    public void changeCards() {
        System.out.println("---------------Change Cards");
    }

    @Override
    public void peekCards() {
        System.out.println("---------------Peek Cards");
    }
}
```

output

```
---------------Play Game
---------------!!!!Not Allowed
---------------!!!!Not Allowed
---------------!!!!Not Allowed
---------------Game Win
---------------Play Game
---------------Double Scores
---------------!!!!Not Allowed
---------------!!!!Not Allowed
---------------Game Win
---------------Play Game
---------------Double Scores
```

```
---------------Change Cards
---------------Peek Cards
---------------Game Lose
---------------Play Game
---------------Double Scores
---------------Change Cards
---------------!!!!Not Allowed
---------------Game Lose
```