4.14

第一次迭代：（完成了 21点的规则实现）

import java.util.ArrayList;  
import java.util.Random;  
import java.util.Scanner;  
  
public class Main {  
 public static void main(String[] args) throws InterruptedException {  
 Scanner scanner = new Scanner(System.*in*);  
 Random random = new Random();  
  
 int playerSum = 0;  
 ArrayList<Object> cards = new ArrayList<>();  
  
 //玩家  
 System.*out*.println("1.抽牌\n2.停牌\n请输入：");  
 int input;  
 int i = 0;//抽牌次数  
 while ((input = scanner.nextInt()) != 2) {  
 i++;  
 int num = random.nextInt(11) + 1;  
 playerSum += num;  
 cards.add(num);  
  
 System.*out*.println("现在的手牌是：");  
 for (Object o : cards) {  
 System.*out*.print(o + " ");  
 }  
 System.*out*.println();  
  
 if (playerSum > 21) {  
 System.*out*.println("爆牌");  
 playerSum = 0;  
 break;  
 }  
 System.*out*.println("1.抽牌\n2.停牌\n请输入：");  
 }  
  
 //庄家  
 int dealerSum = 0;  
 ArrayList<Object> dealerCards = new ArrayList<>();  
 while (true) {  
 int dealerNum = random.nextInt(11) + 1;  
 dealerCards.add(dealerNum);  
 dealerSum += dealerNum;  
  
 System.*out*.println("庄家当前手牌：");  
 for (Object o : dealerCards) {  
 System.*out*.print(o + " ");  
 }  
 System.*out*.println();  
  
 if (dealerSum > 17) {  
 System.*out*.println("庄家停止抽牌");  
 if (dealerSum > 21) {  
 System.*out*.println("庄家爆牌");  
 dealerSum = 0;  
 }  
 break;  
 }  
 Thread.*sleep*(1000);  
 }  
  
 //开始比较  
 if (i == 7) {  
 System.*out*.println("玩家获胜");  
 } else if (playerSum > dealerSum) {  
 System.*out*.println("玩家获胜");  
 } else if (playerSum == dealerSum) {  
 System.*out*.println("平局");  
 } else {  
 System.*out*.println("庄家获胜");  
 }  
 }  
}

第二次迭代：（将数字变为扑克牌显示，同时实现A能作为1或11）

import java.util.ArrayList;

import java.util.Random;

import java.util.Scanner;

public class Main {

public static void main(String[] args) throws InterruptedException {

Scanner scanner = new Scanner(System.in);

Random random = new Random();

String[] model = new String[]{"A", "2", "3", "4", "5", "6", "7", "8", "9", "10", "J", "Q", "K"};

int playerSum = 0;

ArrayList<Object> cards = new ArrayList<>();

//玩家

System.out.println("1.抽牌\n2.停牌\n请输入：");

int input;

int i = 0;//抽牌次数

while ((input = scanner.nextInt()) != 2) {

int num = random.nextInt(12);

cards.add(model[num]);

//统计A的个数，顺便输出现有的手牌

int Anum = 0;

System.out.println("现在的手牌是：");

for (Object o : cards) {

System.out.print(o + " ");

if (o.equals("A")) {

Anum++;

}

}

System.out.println();

//处理手牌数据

playerSum = 0;

for (Object o : cards) {

if (o.equals("J") || o.equals("Q") || o.equals("K")) {

playerSum += 10;

} else if (!(o.equals("A"))) {

playerSum += Integer.parseInt((String) o);

}

}

if (Anum == 1) {

if (playerSum < 11) {

playerSum += 11;

} else {

playerSum++;

}

} else {

playerSum += Anum;

}

if (playerSum > 21) {

System.out.println("爆牌");

playerSum = 0;

break;

}

System.out.println("1.抽牌\n2.停牌\n请输入：");

}

//庄家  
        int dealerSum = 0;  
        ArrayList<Object> dealerCards = new ArrayList<>();  
        while (true) {  
            int dealerNum = random.nextInt(12);  
            dealerCards.add(model[dealerNum]);  
  
            int AAcount = 0;  
            System.out.println("庄家当前手牌：");  
            for (Object o : dealerCards) {  
                System.out.print(o + " ");  
                if (o.equals("A")) {  
                    AAcount++;  
                }  
            }  
            System.out.println();  
  
            //处理庄家手牌数据  
            dealerSum = 0;  
            for (Object o : dealerCards) {  
                if (o.equals("J") || o.equals("Q") || o.equals("K")) {  
                    dealerSum += 10;  
                } else if (!(o.equals("A"))) {  
                    dealerSum += Integer.parseInt((String) o);  
                }  
            }  
            if (AAcount == 1) {  
                if (dealerSum < 11) {  
                    dealerSum += 11;  
                } else {  
                    dealerSum++;  
                }  
            } else {  
                dealerSum += AAcount;  
            }  
  
            if (dealerSum > 17) {  
                System.out.println("庄家停止抽牌");  
                if (dealerSum > 21) {  
                    System.out.println("庄家爆牌");  
                    dealerSum = 0;  
                }  
                break;  
            }  
            Thread.sleep(1000);  
        }

//开始比较

if (i == 7) {

System.out.println("玩家获胜");

} else if (playerSum > dealerSum) {

System.out.println("玩家获胜");

} else if (playerSum == dealerSum) {

System.out.println("平局");

} else {

System.out.println("庄家获胜");

}

}

}

第三次迭代：（增加了规则介绍，通过永真循环实现能一局一局玩下去）

public class Main {  
 public static void main(String[] args) throws InterruptedException {  
 System.*out*.println("规则介绍：\n" +  
 "由玩家首先抽牌，可以随时选择停牌。\n" +  
 "A可以视作1点或者11点。\n" +  
 "如果抽到21点以上则视为爆牌，会自动停牌\n" +  
 "庄家抽牌总和达到17点或以上时会停牌\n" +  
 "最后比较大小，点数更大并且没有爆牌的一方获胜。\n" +  
 "若玩家连续抽到第7张牌并且没有爆牌，那么玩家获胜。");  
 int round = 0;  
 while (true) {  
 round++;  
 System.*out*.println("第" + round + "局");  
 ······  
 }  
 }  
}

完成了21点的基础（核心）功能

遇到问题：A可以视作1或11

解决方案：可以每次循环都遍历一遍当前手牌，统计当前手牌除了A的其他牌的总和，最后根据情况决定A为1还是11；每次循环都重置手牌总和。

4.15

完成了玩家手牌达到21点时自动停牌，优化了当抽牌次数达到7次后自动获胜部分的代码。

70 else if (playerSum == 21) {//玩家21点自动停牌  
 break;  
 } else if (i == 7) {  
 isSeven = 1;  
 break;  
 }

84 if (isSeven == 1) {  
 System.*out*.println("玩家已抽到7张牌");  
 break;  
 }

133 if (isSeven == 1) {  
 System.*out*.println("\n玩家获胜\n");  
 }

4.16

增加下注功能，加入BLACK JACK 1.5倍回报；增加输入的若不是数字要求玩家重新输入；更改规则：庄家会先抽一张牌，玩家会被自动分配两张牌

优化了代码结构

import java.util.ArrayList;  
import java.util.Random;  
import java.util.Scanner;  
  
public class Main {  
 public static void main(String[] args) throws Exception {  
 System.*out*.println("规则介绍：\n" +  
 "每局开始时，庄家会先抽一张牌。\n" +  
 "然后玩家会被自动分配两张牌，之后可以选择继续抽牌或停牌。\n" +  
 "A可以视作1点或者11点。\n" +  
 "如果抽到21点以上则视为爆牌，会自动停牌\n" +  
 "庄家抽牌总和达到17点或以上时会停牌\n" +  
 "最后比较大小，点数更大并且没有爆牌的一方获胜。\n" +  
 "若玩家连续抽到第7张牌并且没有爆牌，那么玩家获胜。\n");  
  
 //赌注  
 int totalMoney = 10000;  
 System.*out*.println("您的初始筹码为：" + totalMoney);  
  
 int round = 0;  
 while (true) {  
 if (totalMoney == 0) {  
 System.*out*.println("您破产了，请下辈子再来");  
 Thread.*sleep*(4000);  
 return;  
 }  
  
 round++;  
 System.*out*.println("第" + round + "局");  
  
 //下注  
 System.*out*.println("您当前剩余筹码：" + totalMoney);  
 System.*out*.println("请输入您想下注的数额：");  
 int money = *Input*();  
 if (money > totalMoney) {  
 System.*out*.println("你没有这么多钱，滚出我们赌场！");  
 Thread.*sleep*(4000);  
 return;  
 } else {  
 System.*out*.println("您本局下注：" + money);  
 }  
 totalMoney -= money;  
  
 Random random = new Random();  
 String[] model = new String[]{"A", "2", "3", "4", "5", "6", "7", "8", "9", "10", "J", "Q", "K"};  
  
 //庄家先抽一张牌  
 ArrayList<Object> dealerCards = new ArrayList<>();  
 int firstNum = random.nextInt(12);  
 dealerCards.add(model[firstNum]);  
 System.*out*.println("庄家当前手牌：");  
 System.*out*.println(model[firstNum]);  
  
 //玩家  
 ArrayList<Object> cards = new ArrayList<>();  
 boolean isSeven = false;//是否抽到7次的指示器  
 boolean isBlackJack = false;//是否为BlackJack的指示器  
  
 System.*out*.println("自动发牌中···");  
 for (int j = 0; j < 2; j++) {  
 int twice = random.nextInt(12);  
 cards.add(model[twice]);  
 }  
 Thread.*sleep*(1500);  
  
 int playerSum = *Player*(cards);  
 if (playerSum == 21) {  
 isBlackJack = true;  
 }  
  
 if (isBlackJack) {  
 System.*out*.println("Black Jack!");  
 } else {  
 System.*out*.println("1.抽牌\n2.停牌\n请输入：");  
 int input = *Input*();  
 int i = 2;//抽牌次数  
  
 while (input != 2) {  
 i++;  
 int num = random.nextInt(12);  
 cards.add(model[num]);  
  
 playerSum = *Player*(cards);  
  
 if (playerSum > 21) {  
 System.*out*.println("爆牌");  
 playerSum = 0;  
 Thread.*sleep*(2000);  
 break;  
 } else if (playerSum == 21) {//玩家21点自动停牌  
 System.*out*.println("21点！");  
 Thread.*sleep*(2000);  
 break;  
 } else if (i == 7) {  
 isSeven = true;  
 break;  
 }  
  
 System.*out*.println("1.抽牌\n2.停牌\n请输入：");  
 input = *Input*();  
 }  
 }  
 System.*out*.println("庄家开始抽牌");  
 Thread.*sleep*(2000);  
  
 //庄家  
 int dealerSum = 0;  
 while (true) {  
 if (isSeven) {  
 System.*out*.println("玩家已抽到7张牌");  
 break;  
 }  
  
 int dealerNum = random.nextInt(12);  
 dealerCards.add(model[dealerNum]);  
  
 int AAcount = 0;  
 System.*out*.println("庄家当前手牌：");  
 for (Object o : dealerCards) {  
 System.*out*.print(o + " ");  
 if (o.equals("A")) {  
 AAcount++;  
 }  
 }  
 System.*out*.println();  
  
 //处理庄家手牌数据  
 dealerSum = 0;  
 for (Object o : dealerCards) {  
 if (o.equals("J") || o.equals("Q") || o.equals("K")) {  
 dealerSum += 10;  
 } else if (!(o.equals("A"))) {  
 dealerSum += Integer.*parseInt*((String) o);  
 }  
 }  
 if (AAcount == 1) {  
 if (dealerSum < 11) {  
 dealerSum += 11;  
 } else {  
 dealerSum++;  
 }  
 } else {  
 dealerSum += AAcount;  
 }  
  
 if (dealerSum > 17) {  
 System.*out*.println("庄家停止抽牌");  
 if (dealerSum > 21) {  
 System.*out*.println("庄家爆牌");  
 dealerSum = 0;  
 } else {  
 System.*out*.println("庄家的手牌总和为：" + dealerSum);  
 }  
 break;  
 }  
 Thread.*sleep*(2000);  
 }  
 Thread.*sleep*(2000);  
  
 //开始比较  
 if (isBlackJack) {  
 System.*out*.println("\n玩家获胜，黑杰克1.5倍奖励！\n");  
 totalMoney += money \* 3;  
 } else if (isSeven) {  
 System.*out*.println("\n玩家获胜\n");  
 totalMoney += money \* 2;  
 } else if (playerSum > dealerSum) {  
 System.*out*.println("\n玩家获胜\n");  
 totalMoney += money \* 2;  
 } else if (playerSum == dealerSum) {  
 System.*out*.println("\n平局\n");  
 totalMoney += money;  
 } else {  
 System.*out*.println("\n庄家获胜\n");  
 }  
 }  
 }  
  
 //防止玩家输入其他字符的输入数字方法  
 static int Input() {  
 Scanner scanner = new Scanner(System.*in*);  
 if (scanner.hasNextInt()) {  
 return scanner.nextInt();  
 } else {  
 System.*out*.println("请输入数字！");  
 System.*out*.print("请输入：");  
 return *Input*();  
 }  
 }  
  
 static int Player(ArrayList<Object> cards) {  
 //统计A的个数，顺便输出现有的手牌  
 int Acount = 0;  
 System.*out*.println("现在的手牌是：");  
 for (Object o : cards) {  
 System.*out*.print(o + " ");  
 if (o.equals("A")) {  
 Acount++;  
 }  
 }  
 System.*out*.println();  
  
 //处理手牌数据  
 int playerSum = 0;  
 for (Object o : cards) {  
 if (o.equals("J") || o.equals("Q") || o.equals("K")) {  
 playerSum += 10;  
 } else if (!(o.equals("A"))) {  
 playerSum += Integer.*parseInt*((String) o);  
 }  
 }  
 if (Acount == 1) {  
 if (playerSum < 11) {  
 playerSum += 11;  
 } else {  
 playerSum++;  
 }  
 } else {  
 playerSum += Acount;  
 }  
  
 return playerSum;  
 }  
}

7月8日

增加了如果玩家输入0则游戏不进行，如果输入三次0就会强制退出游戏的功能（算是修bug？）；

创建gitee远程仓库并且通过git将代码推送上去开源：

https://gitee.com/springbootlearningaccount/black-jack

//赌注  
int totalMoney = 10000;  
System.*out*.println("您的初始筹码为：" + totalMoney);  
int zeroCount = 0;//玩家输入0赌注计数器  
  
int round = 0;  
while (true) {  
 if (totalMoney == 0) {  
 System.*out*.println("您破产了，请下辈子再来");  
 Thread.*sleep*(4000);  
 return;  
 }  
  
 round++;  
 System.*out*.println("第" + round + "局");  
  
 //下注  
 System.*out*.println("您当前剩余筹码：" + totalMoney);  
 System.*out*.println("请输入您想下注的数额：");  
 int money;  
 while ((money = *Input*()) == 0) {  
 if (zeroCount == 3) {//如果玩家输入3次0赌注，游戏直接结束  
 System.*out*.println("不想玩就滚！");  
 Thread.*sleep*(1000);  
 return;  
 }  
 System.*out*.println("请下注！");  
 zeroCount++;  
 }  
 if (money > totalMoney) {  
 System.*out*.println("你没有这么多钱，滚出我们赌场！");  
 Thread.*sleep*(4000);  
 return;  
 } else {  
 System.*out*.println("您本局下注：" + money);  
 }  
 totalMoney -= money;

为项目添加Maven进行打包

pom.xml：

<?xml version="1.0" encoding="UTF-8"?>  
<project xmlns="http://maven.apache.org/POM/4.0.0"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">  
 <modelVersion>4.0.0</modelVersion>  
  
 <groupId>org.example</groupId>  
 <artifactId>BlackJack</artifactId>  
 <version>1.0-SNAPSHOT</version>  
  
 <properties>  
 <maven.compiler.source>8</maven.compiler.source>  
 <maven.compiler.target>8</maven.compiler.target>  
 <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  
 </properties>  
  
 <build>  
 <plugins>  
 <plugin>  
 <groupId>org.apache.maven.plugins</groupId>  
 <artifactId>maven-compiler-plugin</artifactId>  
 <configuration>  
 <source>1.8</source>  
 <target>1.8</target>  
 <encoding>UTF-8</encoding>  
 </configuration>  
 </plugin>  
 <plugin>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-maven-plugin</artifactId>  
 <version>2.6.13</version>  
 <configuration>  
 <mainClass>org.mtz.Main</mainClass>  
 <skip>false</skip>  
 </configuration>  
 <executions>  
 <execution>  
 <id>repackage</id>  
 <goals>  
 <goal>repackage</goal>  
 </goals>  
 </execution>  
 </executions>  
 </plugin>  
 </plugins>  
 </build>  
</project>

第四次迭代：（将重复代码块聚合成一个方法；将处理手牌数据的两个循环合并成一个；将所有重复调用的方法集成到工具类中，优化了代码的格式）

Main：

package org.mtz;  
  
import java.util.ArrayList;  
import java.util.Random;  
  
import static org.mtz.Tools.\*;  
  
public class Main {  
 public static void main(String[] args) throws Exception {  
 System.*out*.println("规则介绍：\n" +  
 "每局开始时，庄家会先抽一张牌。\n" +  
 "然后玩家会被自动分配两张牌，之后可以选择继续抽牌或停牌。\n" +  
 "花牌都视作10点，A可以视作1点或者11点。\n" +  
 "如果抽到21点以上则视为爆牌，会自动停牌。\n" +  
 "庄家抽牌总和达到17点或以上时会停牌。\n" +  
 "最后比较大小，点数更大并且没有爆牌的一方获胜。\n" +  
 "若玩家连续抽到第7张牌并且没有爆牌，那么玩家获胜。\n" +  
 "自动发牌阶段达到21点为黑杰克，会获得1.5倍奖励。");  
  
 //赌注  
 int totalMoney = 10000;  
 System.*out*.println("您的初始筹码为：" + totalMoney);  
 int zeroCount = 0;//玩家输入0赌注计数器  
  
 int round = 0;  
 while (true) {  
 if (totalMoney == 0) {  
 System.*out*.println("您破产了，请下辈子再来");  
 Thread.*sleep*(4000);  
 return;  
 }  
  
 round++;  
 System.*out*.println("第" + round + "局");  
  
 //下注  
 System.*out*.println("您当前剩余筹码：" + totalMoney);  
 System.*out*.println("请输入您想下注的数额：");  
 int money;  
 while ((money = *Input*()) == 0) {  
 zeroCount++;  
 if (zeroCount == 3) {//如果玩家输入3次0赌注，游戏直接结束  
 System.*out*.println("不想玩就滚！");  
 Thread.*sleep*(1000);  
 return;  
 }  
 System.*out*.println("请下注！");  
 }  
 if (money > totalMoney) {  
 System.*out*.println("你没有这么多钱，滚出我们赌场！");  
 Thread.*sleep*(4000);  
 return;  
 } else {  
 System.*out*.println("您本局下注：" + money);  
 }  
 totalMoney -= money;  
  
 Random random = new Random();  
 String[] model = new String[]{"A", "2", "3", "4", "5", "6", "7", "8", "9", "10", "J", "Q", "K"};  
  
 //庄家先抽一张牌  
 ArrayList<Object> dealerCards = new ArrayList<>();  
 int firstNum = random.nextInt(12);  
 dealerCards.add(model[firstNum]);  
 System.*out*.println("庄家当前手牌：");  
 System.*out*.println(model[firstNum]);  
  
 //玩家  
 ArrayList<Object> cards = new ArrayList<>();  
 boolean isSeven = false;//是否抽到7次的指示器  
 boolean isBlackJack = false;//是否为BlackJack的指示器  
  
 System.*out*.println("自动发牌中···");  
 for (int j = 0; j < 2; j++) {  
 int twice = random.nextInt(12);  
 cards.add(model[twice]);  
 }  
 Thread.*sleep*(1500);  
  
 //处理玩家手牌数据  
 int playerSum = *DealSum*(cards);  
 System.*out*.println("现在的手牌是：");  
 *PrintCards*(cards);  
  
 if (playerSum == 21) {  
 isBlackJack = true;  
 }  
  
 if (isBlackJack) {  
 System.*out*.println("Black Jack!");  
 } else {  
 System.*out*.println("1.抽牌\n2.停牌\n请输入：");  
 int input = *Input*();  
 int i = 2;//抽牌次数  
  
 while (input != 2) {  
 i++;  
 int num = random.nextInt(12);  
 cards.add(model[num]);  
 *PrintCards*(cards);  
  
 playerSum = *DealSum*(cards);  
  
 if (playerSum > 21) {  
 System.*out*.println("爆牌");  
 playerSum = 0;  
 Thread.*sleep*(2000);  
 break;  
 } else if (playerSum == 21) {//玩家21点自动停牌  
 System.*out*.println("21点！");  
 Thread.*sleep*(2000);  
 break;  
 } else if (i == 7) {  
 isSeven = true;  
 break;  
 }  
  
 System.*out*.println("1.抽牌\n2.停牌\n请输入：");  
 input = *Input*();  
 }  
 }  
 System.*out*.println("庄家开始抽牌");  
 Thread.*sleep*(2000);  
  
 //庄家  
 int dealerSum = 0;  
 while (true) {  
 if (isSeven) {  
 System.*out*.println("玩家已抽到7张牌");  
 break;  
 }  
  
 int dealerNum = random.nextInt(12);  
 dealerCards.add(model[dealerNum]);  
  
 System.*out*.println("庄家当前手牌：");  
 for (Object o : dealerCards) {  
 System.*out*.print(o + " ");  
 }  
 System.*out*.println();  
  
 //处理庄家手牌数据  
 dealerSum = *DealSum*(dealerCards);  
  
 if (dealerSum >= 17) {  
 System.*out*.println("庄家停止抽牌");  
 if (dealerSum > 21) {  
 System.*out*.println("庄家爆牌");  
 dealerSum = 0;  
 } else {  
 System.*out*.println("庄家的手牌总和为：" + dealerSum);  
 }  
 break;  
 }  
 Thread.*sleep*(2000);  
 }  
 Thread.*sleep*(2000);  
  
 //开始比较  
 if (isBlackJack) {  
 System.*out*.println("\n玩家获胜，黑杰克1.5倍奖励！\n");  
 totalMoney += money \* 3;  
 } else if (isSeven) {  
 System.*out*.println("\n玩家获胜\n");  
 totalMoney += money \* 2;  
 } else if (playerSum > dealerSum) {  
 System.*out*.println("\n玩家获胜\n");  
 totalMoney += money \* 2;  
 } else if (playerSum == dealerSum) {  
 System.*out*.println("\n平局\n");  
 totalMoney += money;  
 } else {  
 System.*out*.println("\n庄家获胜\n");  
 }  
 }  
 }  
}

Tools：

package org.mtz;  
  
import java.util.ArrayList;  
import java.util.Scanner;  
  
public class Tools {  
 public static int Input() {  
 //防止玩家输入其他字符的输入数字方法  
 Scanner scanner = new Scanner(System.*in*);  
 if (scanner.hasNextInt()) {  
 return scanner.nextInt();  
 } else {  
 System.*out*.println("请输入数字！");  
 System.*out*.print("请输入：");  
 return *Input*();  
 }  
 }  
  
 public static int DealSum(ArrayList<Object> cards) {  
 //处理手牌数据  
 int Acount = 0;  
 int playerSum = 0;  
  
 for (Object o : cards) {  
 if (o.equals("J") || o.equals("Q") || o.equals("K")) {  
 playerSum += 10;  
 } else if (!(o.equals("A"))) {  
 playerSum += Integer.*parseInt*((String) o);  
 } else if (o.equals("A")) {  
 Acount++;  
 }  
 }  
 if (Acount == 1) {  
 if (playerSum < 11) {  
 playerSum += 11;  
 } else {  
 playerSum++;  
 }  
 } else {  
 playerSum += Acount;  
 }  
  
 return playerSum;  
 }  
  
 public static void PrintCards(ArrayList<Object> cards) {  
 //输出玩家当前手牌  
 System.*out*.println("现在的手牌是：");  
 for (Object o : cards) {  
 System.*out*.print(o + " ");  
 }  
 System.*out*.println();  
 }  
}