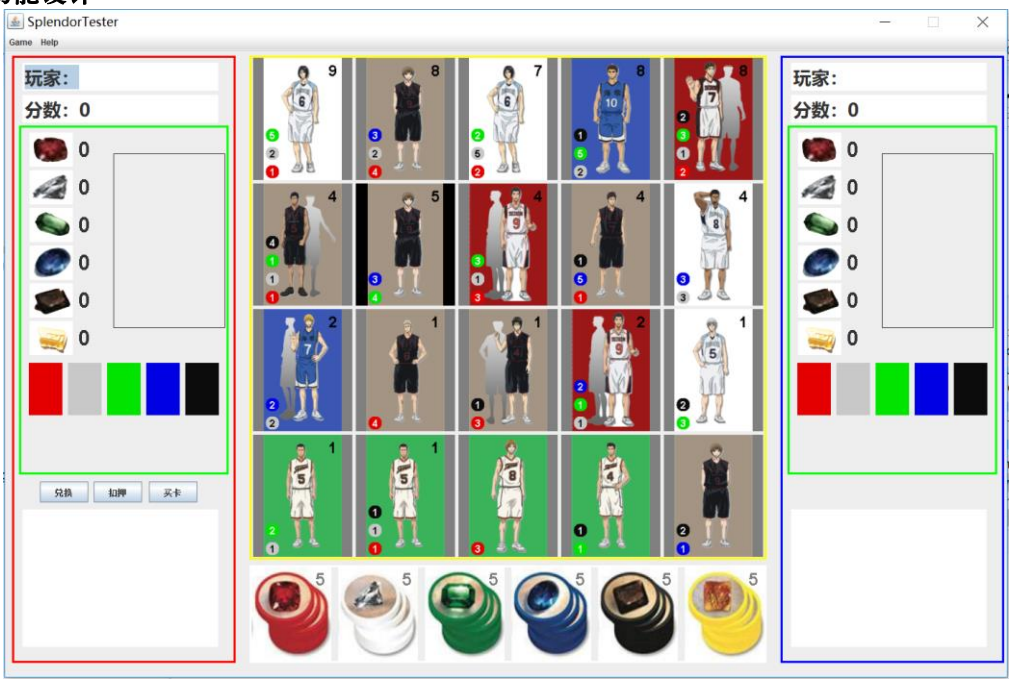
一、功能设计



如图，左右两侧为玩家区，中间部分上侧为牌堆，下侧为币堆。每个玩家每回合可以选择拿硬币（兑换）、押牌（扣押）、买卡其中任一操作。所以需要实现以下功能：

玩家区：
(1) 实时更新玩家的分数
(2) 在下方的消息框中显示玩家执行的操作
(3) 更新玩家战利品区的显示（图中绿框部分）

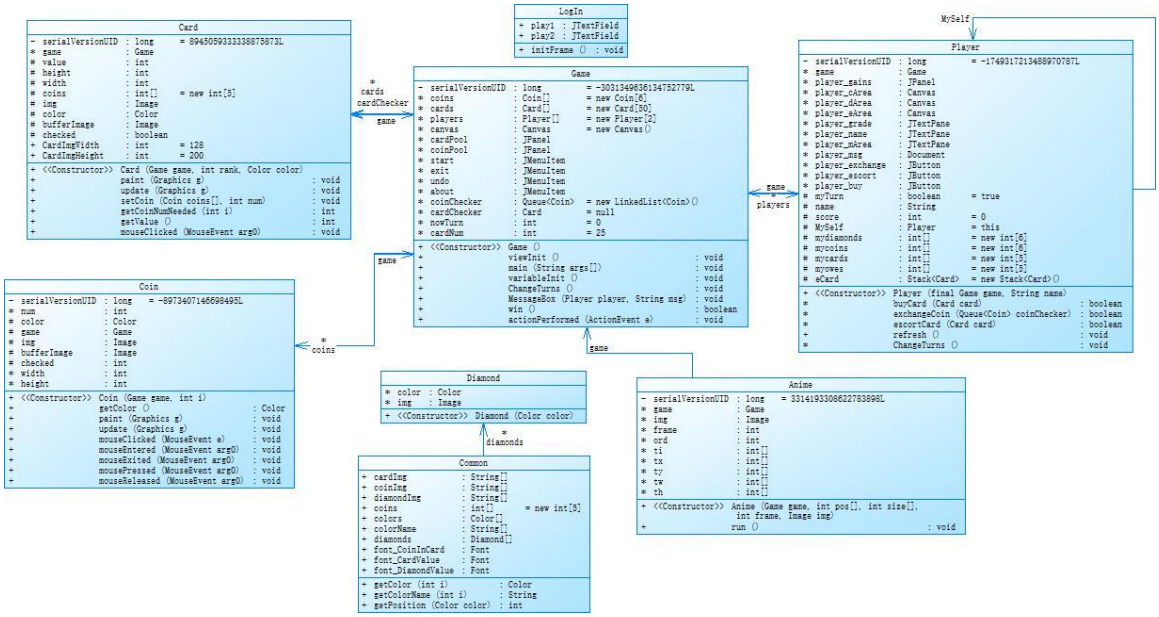
牌堆区：
(1) 记录每张显示出的卡牌的分值和兑换所需硬币数，以判断玩家是否能够进行兑换或扣押
(2) 显示每张牌是否被选中。注意任意时刻牌堆中被选中的牌数至多为一张
(3) 玩家成功买卡时显示卡片飞向玩家区的动态效果，并更新牌堆区

币堆区：
(1) 记录每种硬币的余额，以实时更新判断玩家选择硬币时的规则（比如某种硬币余额<3 就不能同时拿两个）
(2) 显示每张牌是否被选中。注意任意时刻最多只能有 3 个不同颜色或 2 个同色硬币被选中
(3) 玩家成功执行任一操作时更新币堆中的硬币数

此外，游戏整体：
(1) 判断胜利条件，追踪游戏进程
(2) 加入存档、读档、撤回操作等功能（待开发）
(3) 加入 AI 对战功能（待开发）

二、类设计

1.类图（可放大查看，或查看文档目录下的 jpg 文件）



2．各类成员说明

(1) Common类：定义一些常用的静态变量和方法

①静态对象：

public static String[] *cardImg* ：记录卡片的bmp文件位置的字符串
public static String[] *coinImg* ：记录硬币的bmp文件位置的字符串
public static String[] *diamondImg* ：记录钻石的bmp文件位置的字符串

public static int[] *coins* = new int[5] ：记录硬币数量的数组。
public static Color[] *colors* ：记录硬币/卡片颜色的数组。
public static String[] *colorName* ：代表颜色的字符串。
public static Diamond[] *diamonds* ：记录不同颜色的钻石类。

public static Font *font_CoinInCard*
public static Font *font_CardValue*
public static Font *font_DiamondValue* ：用于绘制硬币数量、卡片分数的字体

②静态方法：

public static Color getColor(int i) ： 获取各数组序号对应颜色
public static String getColorName(int i) ： 获取各数组序号对应颜色名字
public static int getPosition(Color color) ： 获取颜色对应的数组序号

(2) Card类：卡片类(继承JLabel)，用于记录牌堆中的卡片信息，实现卡片的消息响应

①对象：

Game game ：游戏类，用于调用游戏中的一些信息，比如当前玩家想要购买的卡片
protected int value ：卡片的分数
protected int height, width ：卡片的宽度，高度（用于绘制）
protected int[] coins = new int[5] ：购买卡片所需要的硬币数
protected Image img ：绘制卡片所调用的图片
protected Color color ：卡片的颜色
protected Image bufferImage ：图片缓存区
protected boolean checked ：表示当前卡片是否处于被选中的状态
public static int *CardImgWidth* = 128
public static int *CardImgHeight*= 200 ：卡片的图片大小为128*200

②方法：

public Card(Game game, int rank, Color color) ：初始化卡片，包括记录卡片信息、得到卡片图片信息、增加监听器等。
public void paint(Graphics g)
public void update(Graphics g) ：绘制卡片的方法

public void setCoin(Coin[] coins, int num) ：设置兑换卡片所需要的各硬币数
public int getCoinNumNeeded(int i) ：得到兑换卡片所需要的指定颜色硬币数
public int getValue() ：获取兑换卡片所需要的指定颜色硬币数

public void mousePressed(MouseEvent arg0) ：设置卡片的消息响应，切换选中/未选中的状态并进行相应的图片更新

(3) Coin类：硬币类(继承JLabael,设置硬币)，用于记录币堆中的硬币信息，实现硬币的消息响应

①对象：

int num ：币堆中某色硬币的数量
protected Color color ：硬币的颜色
protected Game game; ：游戏类，用于调用游戏中的一些信息，比如当前玩家想要兑换的硬币
protected Image img ：硬币图片
protected Image bufferImage ：图片缓存
protected int checked ：判断硬币被选中的次数，0代表未选中，1代表选中一次，2代表选中两次（即兑换两枚同色硬币）
int width, height; ：硬币图像的尺寸

②方法：

public Coin(Game game, int i) ：初始化硬币，包括记录硬币信息、得到硬币图片、增加鼠标监听器等。
public Color getColor() ：得到硬币的颜色
public void paint(Graphics g)
public void update(Graphics g) ：绘图方法，根据硬币被选中的次数（checked的值）

进行相应绘图

`public void mouseClicked(MouseEvent e)` ：设置币堆的消息响应，根据游戏规则判断如何切换币堆状态并进行相应的图片更新

（4）Player 类：玩家类(继承 Jpanel,设置玩家区)，用于记录玩家信息，实现玩家的游戏操作

①对象：

Game game； 游戏类，用于调用游戏中的一些信息，比如判断当前回合的玩家

JPanel player_gains	：战利品区，包括玩家获取的卡牌、硬币等
Canvas player_cArea	：已购卡牌存放区
Canvas player_dArea	：已有钻石（硬币）存放区
Canvas player_eArea	：扣押卡牌存放区
JTextPane player_grade	：玩家分数显示区
JTextPane player_name	：玩家姓名显示区
JTextPane player_mArea	：消息显示区（如执行的操作是否成功等）
Document player_msg	：消息区的内容
JButton player_exchange	：兑换硬币按钮
JButton player_escort	：扣押卡牌按钮
JButton player_buy	：购买卡牌按钮

`protected boolean myTurn = true`	：判断是否是该玩家的回合
`protected String name`	：该玩家的姓名
`protected int score`	：该玩家的分数
`protected Player MySelf = this`	：该玩家
`protected int[] mydiamonds = new int[6]`	：该玩家兑换的各硬币数
`protected int[] mycoins = new int[6]`	：该玩家的硬币+已有卡所替代宝石数(每张购买的卡代表一个永久有效的宝石)
`protected int[] mycards = new int[5]`	：该玩家购买的各色卡牌数量
`protected int[] myowes = new int[5]`	：赎回扣押的卡所需要的硬币数量
`protected Stack<Card> eCard = new Stack<Card>()`	：已扣押（欲赎回）的卡片

②方法：

`public Player(final Game game, String name)` ： 初始化玩家信息，并继承Canvas方法进行战利品区的各子区域的绘制

`boolean buyCard(Card card)`	：买卡操作（含动画效果）
`boolean exchangeCoin(Queue<Coin> coinChecker)`	：拿硬币操作
`boolean escortCard(Card card)`	：扣押/赎回卡片操作（含动画效果）

`public void refresh()` ：刷新玩家区

`void ChangeTurns()` ：判断当前是否是自己的回合，不是的话就不显示按钮（即无法执行任何操作）

(5) Diamond 类：钻石类，用于格式化玩家区的硬币（钻石）信息

`public Diamond(Color color)` 得到指定颜色的钻石图片

(6) Anime类： 设置买牌、押牌动画效果

```
public class Anime extends Canvas implements Runnable {
    private static final long serialVersionUID = 3314193308622783898L;
    Game game                    :游戏类（用于获取canvas）
    Image img                    :要绘制的图片
    int frame, ord               : 动画帧数
    int[] ti, tx, ty, tw, th;    : 不同时刻sleep的时长和图像的位置、大小

    public Anime(Game game, int[] pos, int[] size, int frame, Image img)
    public void run()            : 用线程进行动画绘制
```

(7) Login类： 开始游戏窗口

```
    public static JTextField play1      : 玩家1姓名输入栏
    public static JTextField play2      : 玩家2姓名输入栏

    public void initFrame()             : 绘制开始游戏界面
```

(8) Game类：继承JFrame,游戏主体框架

①对象：

```
    Coin[] coins = new Coin[6]                  : 币堆
    Card[] cards = new Card[50]                 : 牌堆
    Player[] players = new Player[2]            : 2个对战玩家
    Canvas canvas = new Canvas();
    JPanel cardPool, coinPool;                  : 牌堆、币堆的面板
    JMenuItem start, exit, undo, about;         : 菜单栏
    Queue<Coin> coinChecker = new LinkedList<Coin>  : 记录当前玩家选中的硬币信息
    Card cardChecker = null                     : 记录当前玩家选中的卡牌信息

    int nowTurn = 0;                            : 代表当前是哪个玩家的回合
    int cardNum = 25;                           : 显示在cardPool面板的牌数为5*5=25张
```

②方法：

```
    public Game()                               : 初始化游戏
    public void viewInit()                      : 显示界面初始化
    public void variableInit()                  : 硬币、卡片、回合等信息初始化

    public void ChangeTurns()                   : 当前玩家执行完操作后，如果没有胜利，则切换
                                                  玩家（通过调用players类的ChangeTurns方法）
    public void MessageBox(Player player, String msg) : 更新消息框信息

    public boolean win()                        : 判断是否有人获胜

    public void actionPerformed(ActionEvent e) : 菜单栏的消息响应
    public static void main(String[] args)      : 主函数，启动游戏
```

3．主要技术难点和算法设计

（1）画图的图片放缩适配问题

最开始考虑到可能需要实现不同显示器分辨率下的版本，窗口大小可调，那么就需要实现不同尺寸下的画布上缩放图片的功能。对图片而言，其长宽比是固定的，但分配给其的画布大小不一定满足比例要求，故先判断应该留白的位置，再选定需要画出图片的矩形区域，并画出即可。考虑到图片在直接缩放的过程中可能会出现信息的丢失和质量的下降，使用了getInstance 的方法中的图像平滑处理手段。

（2）逻辑的简化和程序的简明化

此游戏相较于普通的五子棋等更为复杂，选牌、兑币、押牌、赎回等操作需要较多逻辑判断和冲突处理，如买不同颜色的硬币最多为三个，但同色的最多选取两个，且每次最多只能兑换一个金色硬币，故在一个简单的点击硬币操作后需要几十上百行的判断代码，为便于程序的逻辑处理和代码的简明，使用了一些数据结构如栈、队列等辅助操作，简化了程序逻辑。

（3）动画的实现和优化

在游戏中有一些炫酷的动画想必更能激发玩家的游戏欲望，因此一开始便考虑加入动画元素。最后使用了一个后台线程在指定时刻的指定路径上画出指定大小的图片，实现了卡片的缩放移动，基本满足了要求。考虑到不同路径距离的移动所需的时间应该不一样，可以先判断移动的距离，再设置移动的帧数和时间，会更为真实。进一步而言，可以加入一些随机因素，形成多种路径，乃至多种速度、加速度加持下的动画路径，会更加有趣，但尚未完成。

（4）设计、实现与封装

此游戏尚未在国内盛行，玩家基数不大，故而也没有成熟的框架可供参考，从界面的设计、到类的实现与封装，全部是慢慢摸索、修改得来。最终仅有Game/Coin/Card/Player/Diamond/ Common 等几个类，但在测试过程中，我们尝试了很多方法，测试了很多界面布局、类的分配情况下的游戏性，最后才主力完善最后一版，颇费心思。

最后程序中 Coin，Card,Player 等类均继承 JPanel 类，在各自的类内实现各自的功能和区域的绘制，然后在主逻辑 Game 类中实现类的组合以及把在各类内绘制好的子面板加入到 Game 类里的主面板中，使得不同类的功能既能独立实现又能很好地互相关联，使得代码逻辑较为清晰，便于修改和完善。