www.onlinedoctranslator.com - naisreP ot hsilgnE morf detalsnarT

پروژه‌نهایی_جاوا

ساخته:فروغ هادی

مقدمه:هدف این پروژه شبیه سازی حالت خاصی از بازی پازل سرنخ است! در این در یکی از اتاق های بازی، یک الماس‌گرانبها پنهان شده است. شما باید اتاق را بدزدید و مکان را حدس بزنید * اجرا به صورت تک نفره خواهد بود. * هیچ‌کد قالبی برای این پروژه ارائه نخواهد شد، پیاده سازی صفر تا صد و نحوه پیشبرد هر مرحله به عهده شماست. استفاده‌از مفاهیم پایه جاوا مانند وراثت علاوه بر این، چند شکلی، انتزاع و کپسوله سازی در پیاده سازی الزامی است. نحوه‌بازی: بازی ما می تواند بین 3 تا 6 شرکت کننده داشته باشد و شامل دو تاس شش ضلعی 21 کاره است. 6 شخصیت:اما، لیام، جک، سوفیا، امیلی، الا 6 مکان: زیر گلدان، کشوی مخفی، پشت عکس، داخل جعبه، زیر میز، بالای‌کمد 9 اتاق: گلخانه، اتاق بیلیارد، مطالعه اتاق، اتاق پذیرایی، اتاق خواب، اتاق پیانو، اتاق غذاخوری، آشپزخانه، کتابخانه.ابتدا هر سه دسته کارت را به صورت تصادفی با هم مخلوط می کنیم و یک کارت از هر عرشه رو به بالا به صورت‌تصادفی جدا کرده و کنار می گذاریم. حالا بقیه کارت ها را با هم ترکیب کنید و آنها را بین شرکا تقسیم کنید آیا آنهارا بازی می کنید؟ با توجه به تعداد شرکت کنندگان، تعداد کارت های آنها متفاوت است D. شرکت کنندگان را به صورت‌تصادفی مرتب کنید و بازی را از اول شخص شروع کنید. نفر اول تاس خود را می اندازد و با توجه به تعداد تاس‌ها وارد یکی از اتاق ها می شود. برای انتخاب اتاق ها، قوانین زیر را رعایت کنید:

-اگر تعداد تاس ها فرد باشد، بازیکن می تواند به اتاق های فرد و اگر عدد زوج باشد، به اتاق های باغ وحش c. - بازیکن‌باید حرکت کند و نمی تواند در اتاقش بماند. - بازیکن حق رفتن به اتاق های مجاور را ندارد. مثلاً اگر بازیکن در اتاق6 باشد، در حرکت بعدی حق ندارد به اتاق های 5 و 7 برود. به محض ورود به اتاق، بازیکن باید بین هر مجموعه ازکارت های مظنون حدس بزند و یک کارت را انتخاب کند. مثال: بازیکن وارد اتاق 5 می شود و جک و بالای کمد را به‌عنوان حدس خود انتخاب می کند و او آن را به دیگران اعلام می کند. با اعلام حدس، این بازیکن از نفر بعدی حرکت می‌کند هر بازیکنی که یکی از سه گزینه را داشته باشد باید به او و کارت خود را برای نمایش اعلام کند. مثال: بازیکن شماره‌یک حدس قبلی را که در مثال انجام دادیم اعلام می کند. بازیکن دو هیچ کدام از کارت ها را ندارد، بنابراین او آن راپاس می کند و بازیکن بعدی چک می کند. فرض کنید بازیکن شماره سه کارت های بالای کشو d و اتاق 5 d را دارد، بنابراین‌یکی از کارت ها به بازیکن یک اعلام می کند. بقیه بازیکنان فقط می دانند که بازیکن شماره سه یکی از سه کارت‌است، اما آنها نمی دانند کدام یک! اگر در پایان یک راند فرد به نتیجه ای رسید که می تواند دزدی کند، مکان و اتاق‌به درستی اگر حدس زد، اعلام می کند که قصد دارد حدس نهایی را انجام دهد، بنابراین از هر کدام سه عدد d را انتخاب‌می کند. اگر حدس درست باشد، بازی تمام شده است و همه از پایان بازی هستند، پاسخ معما را خواهند دانست.اما اگر حدس اشتباه باشد، این بازیکن خارج از بازی است، او دیگر حق پرتاب تاس و حدس زدن را ندارد و فقط‌در صورتی که کارت های سوال داشته باشد، دیگران آن را به فرد مورد نظر نشان می دهند. پس از آن، بازی نفر بعدی‌به همین ترتیب ادامه می یابد. توجه: توجه کنید به محض اینکه یکی از کارت های سوال در دست یکی از بازیکنان‌قرار گرفت و آن را به سوال کننده نشان دهید، این دور ادامه پیدا نمی کند و وارد قسمت بعدی بازی شوید ماحدس نهایی یا بدون حدس برای ادامه بازی خواهیم بود. بازی توسط نفر بعدی پیاده سازی: برای سادگی، شما این بازی‌را شبیه سازی می کنید که یک بازیکن واقعی حضور دارد و بقیه بازیکنان به صورت تصادفی عمل می کنند و حق حدس‌زدن ندارند. یعنی بقیه بازیکنان تاس می اندازند و به طور تصادفی به یکی از اتاق هایی که می توانند می روند و به‌صورت تصادفی به یک سوال پاسخ می دهند. شما باید به بازی با آنها ادامه دهید و معما را حل کنید. برای پیاده سازی‌برای کار با اعداد و آرایه های تصادفی و برقراری ارتباط بین آرایه ها به قسمت Bonus نیاز دارید: در این قسمت‌سعی کنید پیاده سازی را به گونه ای انجام دهید که فقط حدس زدن مختص یک بازیکن نباشد و حداقل یک بازیکن‌دیگر بتواند حدس بزند. . نکات: *ارائه گزارش کامل الزامی است

تحویل‌پروژه گزارش شما باید شامل چالش ها و نحوه حل آنها، جزئیات پیاده سازی و توضیحات برای هر قسمت از کدBe باشد. * کد شما باید بدون هیچ خطایی کامپایل و اجرا شود. * کد شما باید از اصول برنامه نویسی شی گرا پیروی‌کند.

پاسخ:

بیاییدتوضیح مفصلی برای هر کلاس در اجرای بازی Clue ارائه دهیم، که هدف آن ها، چالش هایی که با آن ها مواجه‌می شوند و نحوه پرداختن به آنها در کد را پوشش می دهد.

Card.java:**هدف**:کارت.نشان دهنده یک کارت در بازی است که می تواند یک شخصیت، مکان یا کارت اتاق‌باشد. هر کارت یک نام مرتبط با آن دارد class

پیاده‌سازی:

```
ثبت‌عمومی‌کارت (نام رشته) {
        @Override
        عمومی‌رشته() {
            برگشت‌نام;
        {

        عمومی‌رشته‌getName() {
            برگشت"";
        {
{
```

توضیح:

- **فیلدهاو سازنده**:کارت‌دارای یک فیلد واحدنام‌و یک سازنده برای مقداردهی اولیه آن. رایgetName متدنام کارت را برمی گرداند.
- **روش‌toString**: لغو می کندtoStringبرای برگرداندن نام کارت که برای رفع اشکال و نمایش اطلاعات‌کارت مفید است.

چالش‌ها:

- اطمینان‌حاصل کنید که نام کارت به درستی مقداردهی اولیه شده است و در طول بازی بدون هیچ گونه استثنای اشاره گرتهی قابل دسترسی است.
- پیاده‌سازی ساده و موثرtoStringروشی برای نمایش واضح اطلاعات کارت

راه‌حل:

- یک‌روش سازنده و دسترسی قوی (getName) برای نام کارت.
- غلبه‌کردtoStringروشی برای بازگرداندن مستقیم نام کارت برای سهولت استفاده.

Deck.java:**هدف:**عرشهکلاس مجموعه ای از را مدیریت می کندکارتاشیاء. به هم زدن و کشیدن کارت اجازهمی دهد.

پیاده‌سازی:

```
واردکردن java.util.ArrayList;
واردکردن java.util.Collections;
واردکردن java.util.List;

ثبت‌عمومی{ )(Deck)List>Card< cards
عمومی‌عرشه{ )(کارت ها List>Card<)
این.کارت ها=جدید;)(کارت ها ArrayList><(
{

خلأعمومی‌بر زدن() {
مجموعه‌ها.بر زدن(کارت ها)
{

عمومی‌کارت‌کارت قرعه کشی() {
برگشت‌کارت ها;)(removeFirst.
{

@Override
عمومی‌فهرست<کارت>کارت ها() {
جدیدبرگرد(><ArrayListکارت ها)
{
{
```

توضیح:

- **فیلدهاو سازنده:**عرشهشامل لیستی ازکارتاشیاء اولیه در سازنده استفاده می کند ArrayListبرای انعطاف پذیری
- **به‌هم زدن:**بر زدنروش با استفاده از عرشه را به هم می زندCollections.shuffle.
- **کارت‌های رسم:**کارت قرعه کشیکارت بالایی را از روی عرشه برمی دارد و برمی گرداند.
- **دسترسی:**کارت های دریافتیلیست کارت های موجود در عرشه را برمی گرداند.

چالش‌ها:

- مدیریتوضعیت عرشه در طول عملیات به هم زدن و کشیدن برای اطمینان از گم شدن یا تکرارنشدن کارت.
- پیاده‌سازی روش های کارآمد برای به هم زدن و کشیدن کارت ها.

راه‌حل:

- استفاده‌شده‌Collections.shuffleبرای به هم زدن برای استفاده از عملکرد داخلی جاوا.
- اجراشدهکارت قرعه کشیروش برداشتن و بازگرداندن کارت ها از انتهای عرشه، حصول اطمینان از ترتیبو عملیات صحیح کارت.

Room.java:**هدف:**اتاقکلاس نشان دهنده یک اتاق در بازی است. هر اتاق یک نام دارد.

پیاده‌سازی:

```
ثبت‌عمومی‌اتاق (نام رشته) {


    @Override
    عمومی‌رشته() {
        برگشت‌نام;
    {

    @Override
    عمومی‌رشته‌نام() {
        برگشت"نام";
    {
{
```

توضیح:

- **فیلدهاو سازنده:**اتاق‌داردنام‌فیلدی که در سازنده مقداردهی اولیه می شود. این اطمینان حاصل می‌کند که نام پوچ یا خالی نیست.
- **گیرنده‌و تنظیم کننده:**getNameنام اتاق را برمی گرداند وsetNameنام اتاق را پس از تأیید اعتبار تنظیم‌می کند.
- **روشtoString**: لغو می کندtoStringبرای برگرداندن نام اتاق

چالش‌ها:

- برای‌جلوگیری از خطاهای منطقی در حین بازی، نام های اتاق های غیر پوچ و غیر خالی را اعمال کنید.
- ارائه‌روش ها (setName،getName) برای دسترسی و تغییر نام اتاق ها با حفظ کپسولاسیون.

راه‌حل:

- اعتبارسنجی سازنده برای ایجاد یک استثنا در صورت نامعتبر بودن نام اتاق اجرا شد.
- روش‌های گیرنده و تنظیم کننده برای دسترسی و به روز رسانی نام اتاق با بررسی های اعتبارسنجی مناسب ارائه‌شده است.
- غلبه‌کردtoStringروشی برای تسهیل اشکال زدایی آسان و نمایش اطلاعات اتاق.

Player.java:**هدف:**بازیکن‌کلاس نشان دهنده یک بازیکن در بازی است. هر بازیکن یک نام، یک اتاق فعلی که در آن‌حضور دارد و لیستی از کارت هایی که در اختیار دارد دارد.

پیاده‌سازی:

```
واردکردن java.util.ArrayList;
واردکردن java.util.List;

کلاس عمومی بازیکن {
    فینال خصوصی رشته نام; فینال خصوصی فهرست
    <کارت> دست; خصوصی اتاق اتاق فعلی;


    عمومی بازیکن(نام رشته) {
        این.نام= نام؛
        این.دست=جدید;()<><ArrayList
    {

    عمومی رشته getName() {
        برگشت نام;
    {

    خلأ عمومی افزودن کارت(کارت کارت) {
        دست;)card(.add
    {

    بولی عمومی hasCard(کارت کارت) {
        برگشت دست;)کارت(.contains
    {

    خلأ عمومی setCurrentRoom(اتاق اتاق) {
        این.اتاق فعلی= اتاق؛
    {

    عمومی اتاق getCurrentRoom() {
        برگشت اتاق فعلی;
    {

{
```

توضیح:

- **فیلدهاو سازنده:** بازیکن دارد نام(نام بازیکن) اتاق فعلی(اتاقی که بازیکن در حال حاضر در آن است) ،وکارت ها(لیست کارت هایی که بازیکن در اختیار دارد).
- **گیرندگانو تنظیم کننده ها:** getName نام بازیکن را برمی گرداند، setCurrentRoom اتاق فعلی بازیکن راتنظیم می کند وgetCurrentRoom آن را برمی گرداند.
- **مدیریت‌کارت:** افزودن کارت یک کارت به دست بازیکن اضافه می کند، hasCard بررسی می کند که آیا بازیکن کارت‌خاصی دارد یا خیر، وکارت های دریافتی همه کارت ها را برمی گرداند

چالش‌ها:

- مدیریت‌وضعیت بازیکن از جمله اتاق فعلی و کارت های آنها در طول چرخه حیات بازی.
- پیاده‌سازی روش های کارآمد برای افزودن کارت، بررسی در اختیار داشتن کارت و بازیابی اطلاعات‌بازیکن.

راه‌حل:

- ازیک لیست استفاده کرد (کارت ها) برای مدیریت پویا کارت های بازیکن.
- برایحفظ وضعیت بازی، روش های دریافت کننده/ستر ساده برای نام بازیکن و اتاق فعلی پیاده سازیشده است.
- روش هایبرای افزودن کارت به دست بازیکن، بررسی در اختیار داشتن کارت و بازیابی همه کارت هایی که بازیکندر اختیار دارد، ارائه شده است.

Dice.java:**هدف**:تاسکلاس یک تاس شش طرفه را شبیه سازی می کند.

پیاده‌سازی:

```
واردکردن java.util.Random;

کلاس‌عمومی‌تاس {
    فینال‌خصوصی‌تصادفی‌تصادفی;

    عمومی‌تاس() {
        تصادفی=جدیدتصادفی();
    }

    بین‌المللی عمومی‌رول() {
        برگشت‌تصادفی(6.nextInt) +1;
    }
}
```

- **فیلدهاو سازنده**:تاس‌داردتصادفی‌فیلد اولیه در سازنده با استفاده از java.util.Random.
- **روش‌رول**:رول‌یک عدد تصادفی بین 1 و 6 تولید می کند و یک تاس را شبیه سازی می کند.

چالش‌ها:

- شبیه‌سازی یک تاس عادلانه و تصادفی برای حرکات بازیکن و سایر مکانیک های بازی.
- اطمینان‌ازرول‌روش به طور مداوم اعداد را در محدوده مشخص شده (1 تا 6) تولید می کند.

راه‌حل:

- اولیه‌شدتصادفی‌استفاده کردن‌java.util.Randomبرای اطمینان از تصادفی بودن در تاس ریزی.
- اجراشدرول‌روشی برای تولید اعداد تصادفی بین 1 و 6، مناسب برای مکانیک بازی های معمولی‌مبتنی بر تاس.

RoomBuilder.java:**هدف**:RoomBuilderکلاس منطق ساختن را در بر می گیرداتاق اشیاءبا پارامترهای اختیاری تضمین می کند که الف‌اتاق‌شی با یک نام معتبر ایجاد می شود که انعطاف پذیری‌را در نحوه نمونه سازی اتاق ها فراهم می کند.

پیاده‌سازی:

کلاس‌عمومی RoomBuilder {

نام‌رشته خصوصی;

Public RoomBuilder)({

this.name = "";

{

{)نام رشته(Public RoomBuilder setName

this.name = نام;

این‌را برگردان؛

{

ساخت‌اتاق عمومی() {

if)name == null || name.isEmpty)(({

;)"نام اتاق نمی تواند خالی یا خالی باشد"(IllegalStateException جدید avêtin

{

بازگشت‌اتاق جدید (نام)؛

{

{

توضیح:

- **زمینه‌های**:RoomBuilderدارای یک فیلد خصوصی واحدنام‌برای حفظ نام اتاق در حال ساخت.
- **سازنده**: اولیه می کندنام‌به یک رشته خالی، که امکان تنظیم نام را از طریقsetNameروش.

- **setNameروش**: تنظیم می کندنامفیلد و برمی گرداندRoomBuilderخود شیء (این). این روش از الگوی‌سازنده پیروی می کند و زنجیره بندی روش را امکان پذیر می کند.
- **ساختن‌روش**: جدید را می سازداتاق‌شی با ارائه شده‌نام. بررسی می کند که آیا نام‌قبل از ایجاد معتبر است (تهی یا خالی نیست).اتاق‌هدف - شی. اگرنام‌نامعتبر است، آن پرتاب‌می کندIllegalStateException.

استفاده:

- بااستفاده ازRoomBuilder، می توانید ایجاد کنیداتاق‌اشیاء به صورت واضح و مختصر:

```
Room Room = New RoomBuilder)(

)"آشپزخانه"( setName .

.ساختن()؛
```

این‌مثال نشان می دهد که چگونهRoomBuilderایجاد را ساده می کنداتاق‌اشیاء با اجازه دادن به شما برای تنظیم‌نام اتاق با استفاده ازsetNameو سپس بسازیداتاق‌شی باساختن.

فواید:

- **انعطاف‌پذیری**: امکان تنظیم پارامترها (در این مورد، نام اتاق) با زنجیره متد را فراهم می کندو کد را خواناتر و گویاتر می کند.
- **اعتبارسنجی**: این را تضمین می کنداتاق‌اشیاء با نام های معتبر با بررسی مقادیر خالی یا خالی قبل‌از نمونه سازی ایجاد می شوند.
- **کپسوله‌سازی**: منطق ساخت و ساز را کپسوله می کنداتاق‌اشیاء در یک کلاس جداگانه (RoomBuilder، تفکیک نگرانی ها و ترویج مسئولیت واحد.

چالش‌ها:

- **مدیریت‌دولتی**: اطمینان از اینکه وضعیت سازنده (نام‌میدان) به درستی در طول فرآیند ساخت‌و ساز مدیریت می شود تا از رفتارهای ناخواسته یا ناسازگاری جلوگیری شود.
- **رسیدگی‌به خطا**: رسیدگی به استثنائات (مانندIllegalStateException) به طور مناسب برای ارائه پیام‌های خطای معنی دار و اطمینان از استحکام در ایجاد شی.

راه‌حل:

- یک‌الگوی سازنده ساده را که در آن روش زنجیره ای (setName) امکان تنظیم ویژگی ها و ساخت اشیاء(ساختن).
- استفاده‌از مدیریت استثنا (IllegalStateException) برای اجرای قوانین مربوط به ساخت شی، مانندا‌طمینان از ورودی معتبر برایاتاق‌نام ها

نتیجه

راRoomBuilderکلاس وضوح و انعطاف پذیری ایجاد را افزایش می دهداتاق‌اشیاء در جاوا، با استفاده از الگوی‌سازنده برای ساده سازی نمونه سازی شی با پارامترهای اختیاری. توسط

بامحصور کردن منطق ساخت و اعتبار در یک کلاس سازنده اختصاصی، کد تمیزتر و قابل نگهداری تر را با رعایت‌اصول طراحی شی گرا ترویج می کند.

این‌گزارش یک نمای کلی ازRoomBuilderکلاس، هدف آن، جزئیات پیاده سازی، مثال های استفاده، مزایا،چالش ها و راه حل ها. این نشان می دهد که چگونه الگوی سازنده ایجاد و مدیریت قوی شیء را در برنامه‌نویسی جاوا تسهیل می کند.

```
قسمت‌آخر این کلاس رو پاک کردم و نوشتمکلاس عمومی{ RoomBuilder

        عمومیRoomBuildersetName() {
            این‌را برگردان;
        {


        عمومیاتاق‌ساختن() {
            باطل‌برگرداند;
        {
{
```

زیرابه این ترتیب کد می تواند بدون اخطار بهتر اجرا شود.

هدف:ClueGame.javaکلاس کل بازی، از جمله مقداردهی اولیه، جریان بازی، چرخش‌بازیکن و شرایط برد را هماهنگ می کند.

پیاده‌سازی:

```
واردکردن   java.util.ArrayList;
واردکردن   java.util.List;
واردکردن   java.util.Random;

کلاس‌عمومی{ ClueGame
    فینال‌خصوصی‌لیست<بازیکن>بازیکنان; فینال خصوصی
    فهرست<اتاق>اتاق ها; خصوصی
                عرشه  شخصیت‌دک;
    خصوصی      عرشه  مکان‌عرشه;
    خصوصی      عرشه  اتاق‌عرشه;
    فینال‌خصوصی‌تاس‌تاس; خصوصی
                کارت  شخصیت‌پنهان;
    خصوصی      کارت  مکان‌پنهان;
    خصوصی      کارت  hiddenroom;
    خصوصی      بولی  بازی‌برد;

    عمومیClueGame(بین المللی{ )numPlayers
        بازیکنان=جدید;)(<ArrayList> اتاق ها=جدید;(
        (<ArrayList> تاس=جدیدتاس()؛ بازی برد=
        نادرست؛)numPlayers(initGame ;


    {

    خلأخصوصیinitGame(بین المللی{ )numPlayers
```

```
// شخصیت ها
List<Card> کاراکترها = List.از(
    جدید Card("اما"),
    جدید Card("لیام"),
    جدید Card("جک"),
    جدید Card("سوفیا"),
    جدید Card("امیلی"),
    جدید Card("الا")
);
شخصیت‌دک = جدید عرشه<Card>(شخصیت ها); شخصیت دک.بر
زدن();

// مکان ها
فهرست<کارت> مکان = فهرست.از(
    جدیدکارت("زیر گلدان") جدیدکارت("کشوی مخفی"),
    جدیدکارت("پشت عکس") جدیدکارت("داخل جعبه"),
    جدیدکارت("زیر میز") جدیدکارت("بالای کمد")
);
مکان‌عرشه = جدید عرشه<Card>(مکان ها); مکان عرشه
.برزدن();

// اتاق ها
List<Card> roomList = فهرست.از(
    جدید Card("گلخانه"),
    جدید Card("اتاق بیلیارد") کارت("اتاق
    جدید مطالعه") کارت("پذیرایی
    جدید                اتاق"),
    جدید Card("اتاق خواب"),
    جدید کارت("پیانو          اتاق"),
    جدید کارت(" ناهار خوری   اتاق"),
    جدید Card("آشپزخانه"),
    جدید Card("کتابخانه")
);
اتاق‌عرشه = جدید عرشه<Card>(لیست اتاق); اتاق عرشه.بر
زدن();

// با استفاده از RoomBuilder اتاق ها را به بازی اضافه کنید برای(
کارت_ : فهرست اتاق ها) {
    ساخت‌اتاق = جدید RoomBuilder().setName(اتاق ها).build().add;

}

// کارت های مخفی را کنار بگذارید
شخصیت‌پنهان = شخصیت دک.کارت قرعه کشی(); =مکان
مکان‌پنهان = عرشه.کارت قرعه کشی(); =اتاق عرشه.کارت قرعه
hiddenroom کشی();

// بازیکنان را مقدار دهی اولیه کنید و کارت ها را پخش کنید
برای(من = 0; من < numPlayers; من++) {
    بازیکنان.اضافه کردن(جدیدبازیکن("بازیکن " + (من + 1)));
}
dealCards();
{
```

```
خلأخصوصیکارت های معامله() {
= allCards List<Card>جدید(addAll.allCards;)
ArrayList<>(شخصیت دک(addAll.allCards;)(مکان.cards)
عرشه(addAll.allCards;)(اتاق عرشه.cards)(;

بین‌المللیبازیکن ایندکس =0; در حالی که{ )
!allCards.isEmpty)(
بازیکنان+ playerIndex = )playerIndex ;)
(1.get)playerIndex(.addCard)allCards.removeFirst) %بازیکنان.اندازه()؛
{
{

خلأعمومیشروع بازی() {
// حلقه اصلی بازی در حالی
که(!بازی برد) {
برای(بازیکن:بازیکنان) {
؛)بازیکن( takeTurn
// بررسی کنید که آیا بازیکنی برنده شده یا بازی ادامه دارد اگر(بازی برد)زنگ تفریح;

{
{
سیستم.بیرون(".println)"بازی تمام شد!")
{

خلأخصوصینوبت(بازیکن) {
بین‌المللیرول =تاس;)(roll.
سیستم.بیرون+ )player.getName)(println".)"غلتید"+ رول)؛

اگر== Room newRoom = getNewRoom)player.getCurrentRoom)(, roll(;
newRoom)خالی) {
سیستم.بیرون+ )player.getName)(println".)"نمی توان به یک جدید حرکت کرد
اتاق.")
برگشت;
{
player.setCurrentRoom)newRoom(;
سیستم.بیرون+ )player.getName)(println".)"حرکت می کند به";)(newRoom.name +

// حدس بزن
کارت‌حدس زده شخصیت =شخصیت دک(.get)(.cards)جدید تصادفی().nextInt(
شخصیت‌دک;)(((.size)(.cards)// حدس تصادفی
کارت‌حدس زده محل =مکان عرشه(.get)(.cards)جدید تصادفی().nextInt(مکان
عرشه;)(((.size)(.cards)// حدس تصادفی
سیستم.بیرون+ )player.getName)(println".)"حدس می زند"+ (
((;"+ newRoom.name)( + "که در "+ guessedCharacter.getName)" با
+ guessedPlace.getName)

برای(بازیکن پلیر دیگر:بازیکنان) {
اگر{ )بازیکن =! OtherPlayer(
|| )otherPlayer.hasCard)guessedCharacter(اگر
otherPlayer.getCurrentRoom)(.name)(.equals)newRoom.name)(((
otherPlayer.hasCard)guessedPlace( || }
// نمایش کارت به بازیکن حدس زدن (شبیه سازی)
سیستم.بیرون+ )otherPlayer.getName)(println" کارتی را نشان می دهد
به";)(player.getName +
برگشت;
{
```

```
{
{

// بررسی کنید که آیا حدس درست است (سناریوی حدس نهایی)
اگر(guessedCharacter.getName)(.equals)شخصیت پنهان)(.getName)                &&
guessedPlace.getName)(.equals)مکان پنهان)(.equals)newRoom.name)( && ))
System.hiddenroom.getName) { ))).getName.بیرون+ )
"بازی را برده است".println)player.getName)
درست‌حدس زدن"+ )guessedCharacter.getName +" که در "+ )newRoom.name + " با ";))
+ guessedPlace.getName)
بازی‌برد=درست است، واقعی;
{
{

خصوصیاتاق getNewRoom)اتاق فعلی اتاق،بین المللی‌رول) {
List>Room< possibleRooms = جدید;)(ArrayList><) برای(اتاق اتاق:
اتاق‌ها) {
بین‌المللی= roomIndex اتاق ها;)room(indexOf.
اگر))رول %2==0&&roomIndex%2==0) || )% رول2=!0&&
RoomIndex %2=!0)) {
اگر== currentRoom)خالی|| ریاضی.عضلات شکم- roomIndex)
اتاق‌ها< ))currentRoom(indexOf.1) {
possibleRooms.add)room(;
{
{
{
اگر))possibleRooms.isEmpty(              }
باطل‌برگرداند;
{
برگشتpossibleRooms.get)جدید          Random)(.nextInt)possibleRooms.size)));
{

خلأاستاتیک عمومیاصلی(رشته[] آرگ) {
بازیClueGame =جدیدبازی سرنخ (4)// مثال با 4 بازیکن ;game.startGame)(

{
{
```

- **زمینه‌های:**ClueGameدارای زمینه هایی برای بازیکنان، اتاق ها، عرشه ها (شخصیت، مکان و اتاق)، تاس، کارت‌های مخفی و یک بولی‌بازی برد.
- **سازنده(ClueGame):** مولفه های بازی را راه اندازی می کند (بازیکنان،اتاق ها،تاس، بازی‌برد) و با تماس بازی را شروع می کندinitGame.
- **initGameروش:** کاراکترها، مکان ها و عرشه های اتاق را تنظیم می کند، آنها را به هم می ریزد، بازیکنان را مقداردهی اولیه می کند،کارت ها را می فروشد، و کارت های مخفی را کنار می گذارد.
- **کارت‌های معامله‌روش:** کارت ها را پس از زدن بین بازیکنان توزیع می کند و وضعیت بازی را مقداردهی اولیه می‌کند.
- **شروع‌بازی‌روش:** حلقه اصلی بازی که از طریق بازیکنان و تماس ها تکرار می شودنوبتبرای هر بازیکن‌تابازی برددرست است.
- **نوبت‌روش:** چرخش بازیکن را با انداختن تاس، حرکت به اتاق جدید، حدس زدن، بررسی کارت‌های بازیکنان دیگر و تعیین برنده شدن بازی شبیه سازی می کند.
- **getNewRoomروش:** اتاق های معتبری را تعیین می کند که بازیکن می تواند بر اساس تاس انداختن و اتاق فعلی‌حرکت کند.
- **اصلی‌روش:** نقطه ورود برای شروع بازی Clue با تعداد مشخصی از بازیکنان.

چالش‌ها:

- حصول‌اطمینان از حرکت بازیکنان بین اتاق ها بر اساس تاس ریزی و قوانین بازی بدون خطا یاناسازگاری.
- رسیدگی‌به حدس های بازیکن، تعامل کارت بین بازیکنان، و تأیید شرایط برد (بازی برد).

راه‌حل:

- روش‌های اجرا شده (getNewRoom،نوبت،شروع بازی) برای مدیریت اقدامات بازیکن، انتقال اتاق وجریان بازی.
- استفاده‌شده‌تاس‌برای تاس های تصادفی وکارت‌کلاس هایی برای مدیریت موثر کارت های بازیکن‌و عناصر بازی.

نتیجه

اجرای‌بازی Clue از اصول شی گرا مانند کپسوله سازی، وراثت و چندشکلی برای مدل سازی موجودیت های‌بازی استفاده می کند.بازیکن،اتاق،کارت،عرشه،تاس) و تعامل آنها. هر کلاس به جنبه های خاصی از مکانیک‌بازی می پردازد و از تفکیک واضح نگرانی ها و قابلیت نگهداری اطمینان حاصل می کند. این پیاده سازی‌مدیریت قوی قوانین بازی، اقدامات بازیکن و شرایط برد را تضمین می کند و شبیه سازی منسجمی‌از بازی تخته ای Clue را ارائه می دهد.

بااستفاده از ویژگی های جاوا و کتابخانه های استاندارد (java.util.Random، java.util.Collections)، پیاده سازی به مکانیک های گیم پلی کارآمد و قابل اعتماد دست می یابد. هرکلاس و روش برای ایفای نقش های خاص در چارچوب بازی طراحی شده است و قابلیت استفاده‌مجدد و توسعه پذیری کد را ارتقا می دهد.

این‌گزارش تصمیمات طراحی، چالش های پیش رو و راه حل های پیاده سازی شده برای ایجاد یک شبیه سازی بازی Clue کاربردی در جاوا را تشریح می کند.

دوکلاس دیگر اما نه اصلی:

Random.java:با فرض اینکه شما یک ایجاد کرده ایدتصادفی.javaکلاسی برای کپسوله سازی تولید اعداد تصادفی برای‌تاس های ریخته شده در بازی سرنخ شما. در اینجا یک پیاده سازی معمولی وجود دارد:

واردکردن java.util.Random;

تصادفی‌ساز کلاس عمومی {

خصوصی‌استاتیک نهایی Random random = new Random();

عمومی static int rollDice(int sides) {

بازگشت random.nextInt(sides) + 1;

}

}

توضیح:

- **هدف**:تصادفی ساز.شما ارائه می دهد Clue یک ابزار ساده برای تولید اعداد تصادفی، به ویژه‌برای شبیه سازی تاس در بازی class
- **پیاده‌سازی**:
  - ازجاوا استفاده می کندتصادفیکلاس برای تولید اعداد تصادفی
  - rollDiceرامتد یک آرگومان می گیردطرفین(تعداد اضلاع روی تاس) و یک عدد صحیح تصادفی‌بین 1 و برمی گرداندطرفین.
  - random.nextInt(sides)یک عدد صحیح تصادفی از 0 تا تولید می کندطرف - 1و اضافه کردن1 تضمین می کند که نتیجه بین 1 وطرفین.

استفاده:

- دراجرای بازی Clue خود می توانید استفاده کنید Randomizer.rollDice(6)برای شبیه سازی چرخاندن یک‌قالب شش وجهی.

فواید:

- **کپسوله‌سازی**: منطق تولید اعداد تصادفی را در یک کلاس اختصاصی کپسوله می کند (تصادفی‌ساز، ارتقاء قابلیت استفاده مجدد و نگهداری کد.
- **سادگی**: یک روش ساده ارائه می دهد (rollDice) برای تولید اعداد تصادفی با پارامترهای مشخص‌شده (طرفین).

IllegalStateException:با فرض اینکه شما ایجاد کرده ایدIllegalStateException.javaکلاس برای رسیدگی به خطاهای‌خاص مربوط به وضعیت بازی در اجرای بازی Clue شما. در اینجا یک مثال اساسی است:

کلاس‌عمومی IllegalStateException RuntimeException { را گسترش می دهد

عمومیIllegalStateException (پیام رشته ای) {

فوق‌العاده (پیام)؛

}

عمومیIllegalStateException (پیام رشته، علت قابل پرتاب) {

فوق‌العاده (پیام، علت)؛

}

{

توضیح:

- **هدف:**IllegalStateExceptionکلاس گسترش می یابدRuntimeExceptionنشان دادن شرایطغیرقانونی یا غیرمنتظره ای که در حین اجرا رخ می دهد.
- **پیاده‌سازی**:
  - سازندگان‌را برای تنظیم یک پیام خطا (پیام) و اختیاری یک علت (علت پرتابی) برای استثنا.
  - رفتارازRuntimeException، آن را برای استثناهای علامت نخورده مناسب می کند که نشان‌دهنده خطاهای برنامه نویسی یا شرایط غیرمنتظره است.

استفاده:

- دراجرای بازی سرنخ خود، ممکن است یک مورد را پرتاب کنیدIllegalStateException هنگام‌مواجهه با حالت های بازی نامعتبر، مانند تلاش برای دسترسی به نام اتاقی که وجود دارد خالییا خالی

فواید:

- **رسیدگی‌به خطا**: با سیگنال دادن به موقعیت های غیرمنتظره که نیاز به توجه فوری دارند، مدیریت‌خطا را تسهیل می کند.
- **پاک‌کردن پیام**: امکان تعیین پیام های خطای آموزنده (پیام) و به صورت اختیاری ردیابی علت(علت) از استثناء.

ادغام‌با بازی سرنخ

- **ادغام**: گنجاندنتصادفی سازبرای تاس ریختن ()6(Randomizer.rollDice) در داخلClueGameکلاس برای‌شبیه سازی حرکات و اقدامات بازیکن.
- **رسیدگی‌به استثنا**: استفاده کنیدIllegalStateExceptionدر داخلClueGameبرای رسیدگی به خطاهای‌مربوط به ناسازگاری حالت بازی، اطمینان از اجرای روان و گزارش خطا.

این‌کلاس ها (تصادفی.javagjava.IllegalStateException) با ارائه ابزارهای ضروری برای تولید اعداد تصادفی و مدیریت‌خطای ساختاریافته، به عملکرد و استحکام شبیه سازی بازی Clue خود کمک کنید. آنها وضوح، قابلیت‌اطمینان و قابلیت نگهداری پایگاه کد شما را افزایش می دهند و با اصول شی گرا همسو می شوند وشیوه های طراحی نرم افزار موثر را ترویج می کنند.