Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и кибербезопасности

Высшая школа «Компьютерных технологий и информационных систем»

**ОТЧЕТ по лабораторной работе №8**

по дисциплине «Системный подход в разработке программного обеспечения»

**Выполнил:**

студент группы 5130902/20201 \_\_\_\_\_\_\_\_\_\_\_\_\_ А. И. Сафонов

подпись

**Проверил:**

кандидат тех. наук, доцент \_\_\_\_\_\_\_\_\_\_\_\_ С. А. Нестеров

подпись

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_ 2024г.

Санкт-Петербург

2024

1. **По уроку 7.5**
2. **Упражнение 1**

Продолжайте редактировать проект PrisonTest. Вам предоставлена версия этой

программы. Создайте класс Cell, который содержит следующее:

– Строка name камеры.

– Значение Boolean, описывающее открытую дверь.

– Конструктор двух аргументов, который устанавливает оба поля.

Измените класс Prisoner следующим образом:

– Содержит поле Cell.

– Устанавливает поле Cell на основе параметра конструктора.

– Выводит name камеры в качестве метода display()

Создайте метод openDoor() в классе Prisoner. Получите доступ и измените

соответствующее поле в объекте Cell следующим образом:

• Если дверь закрыта, откройте ее.

• Если дверь открыта, закройте ее.

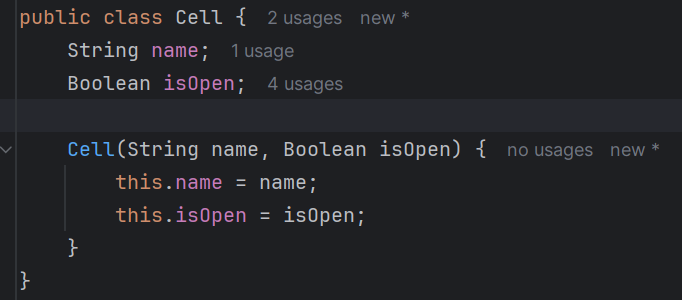
Выведите, открывается или закрывается ли дверь.

В главном методе:

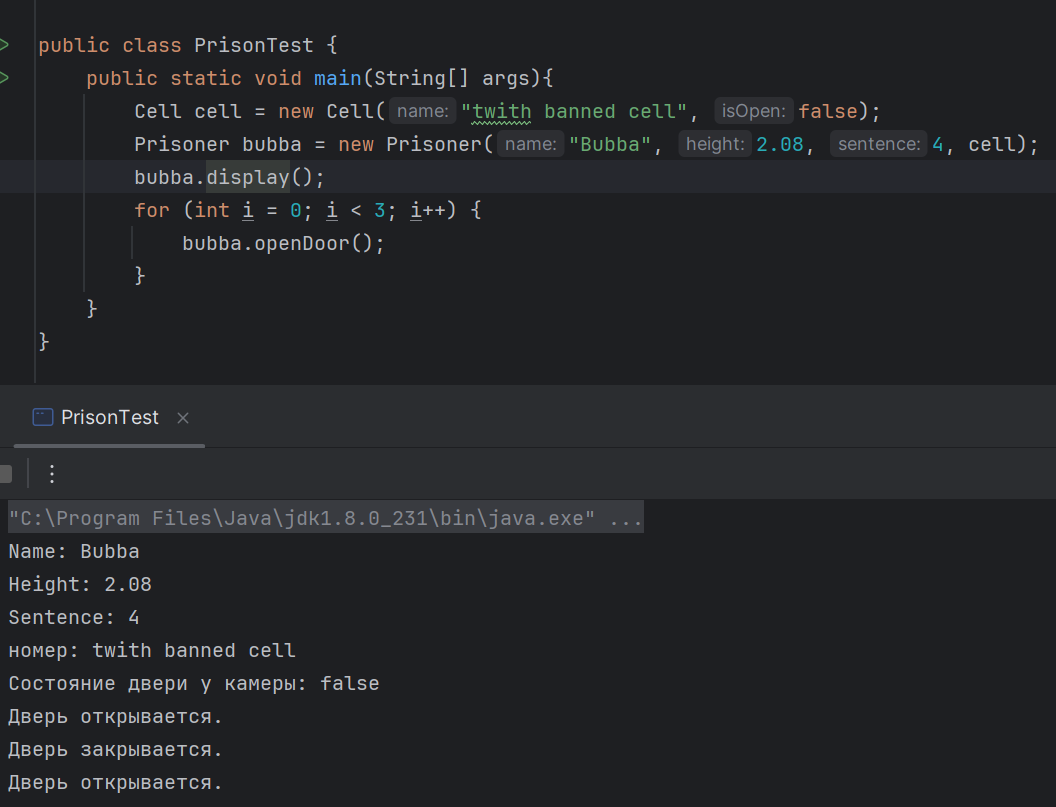
– Создание экземпляров объектов Cell и Prisoner.

– Вызовите метод display() заключенного один раз.

– Вызовите метод openDoor() несколько раз.

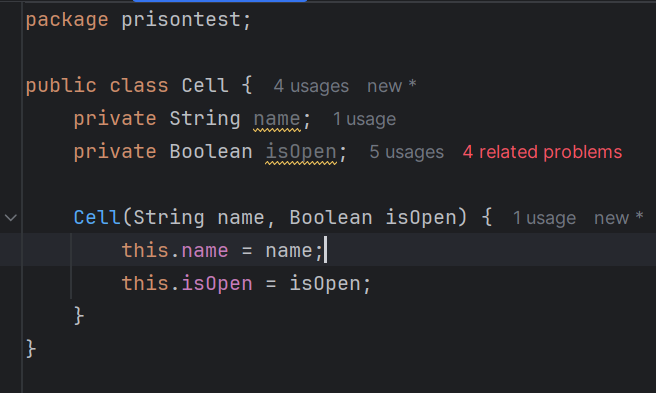
****

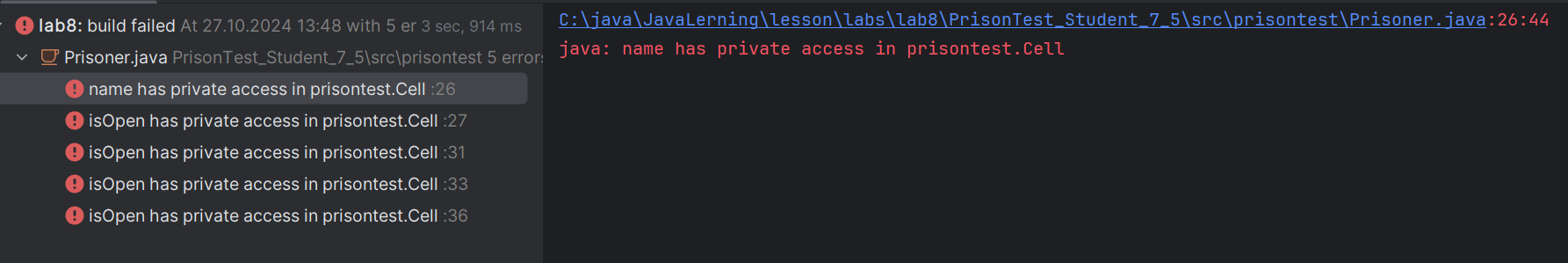
****

****

1. **Упражнение 2**

* Продолжайте редактировать проект PrisonTest.
* Изменение класса Cell:
* Измените значение его полей на значение private.
* Сохраните файл.
* Содержит ли NetBeans какие-либо неполадки?
* Что это за неполадки?
* Где они происходят?

****

****

Ошибка возникает из-за того, что доступ к полям private не может быть

получен за пределами класса Cell.

1. **Упражнение 3**

Продолжайте редактировать проект PrisonTest. Измените класс Cell следующим

образом…

– Устанавливающие методы существуют для полей name и isOpen.

– Имеется поле частного кода безопасности из 4-х цифр. Оно

инициализируется из конструктора и не имеет метода чтения.

–

Существует устанавливающий метод для открытия/закрытия выполняет следующие действия:

* Принимает код безопасности как аргумент.
* Выведите, если код неверен.
* Если код верен и дверь закрыта, открывает ее.
* Если код верен и дверь открыта, закрывает ее.
* Выводит, если дверь открыта или закрыта.

Измените класс Prisoner следующим образом…

– Метод display() выводит имя камеры.

– Метод openDoor() удален.

Измените главный метод следующим образом…

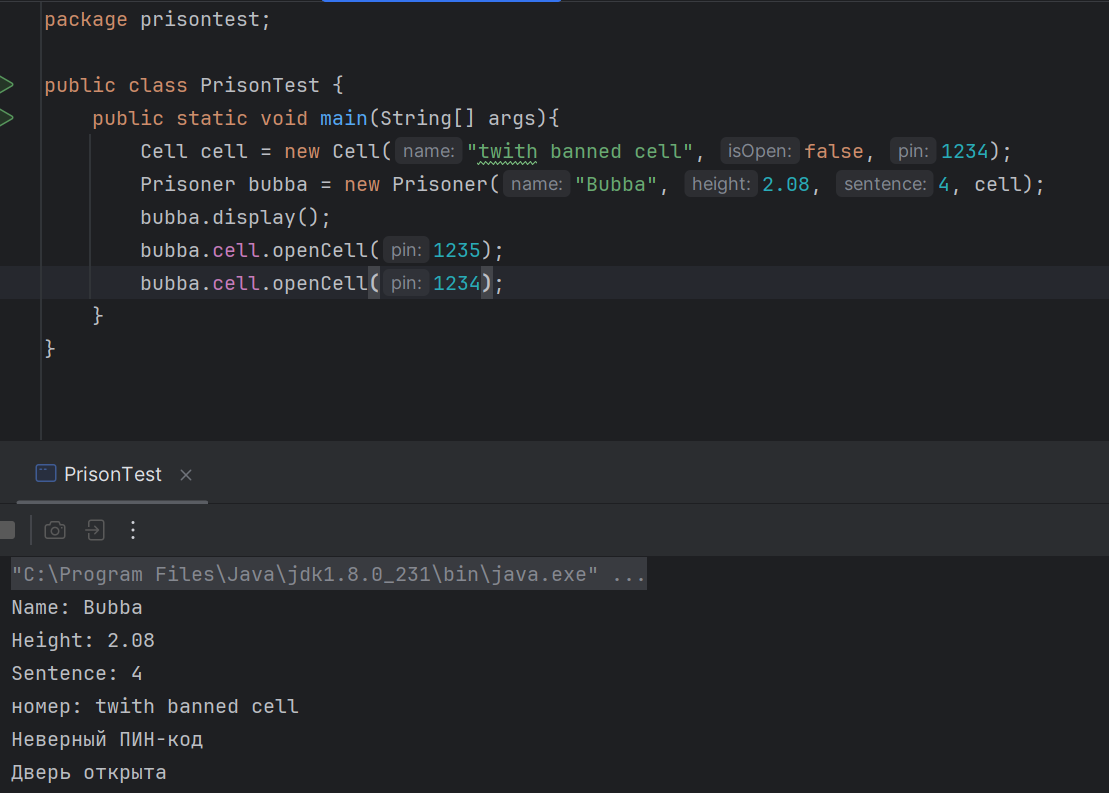
– Для объекта Cell экземпляр создан надлежащим образом.

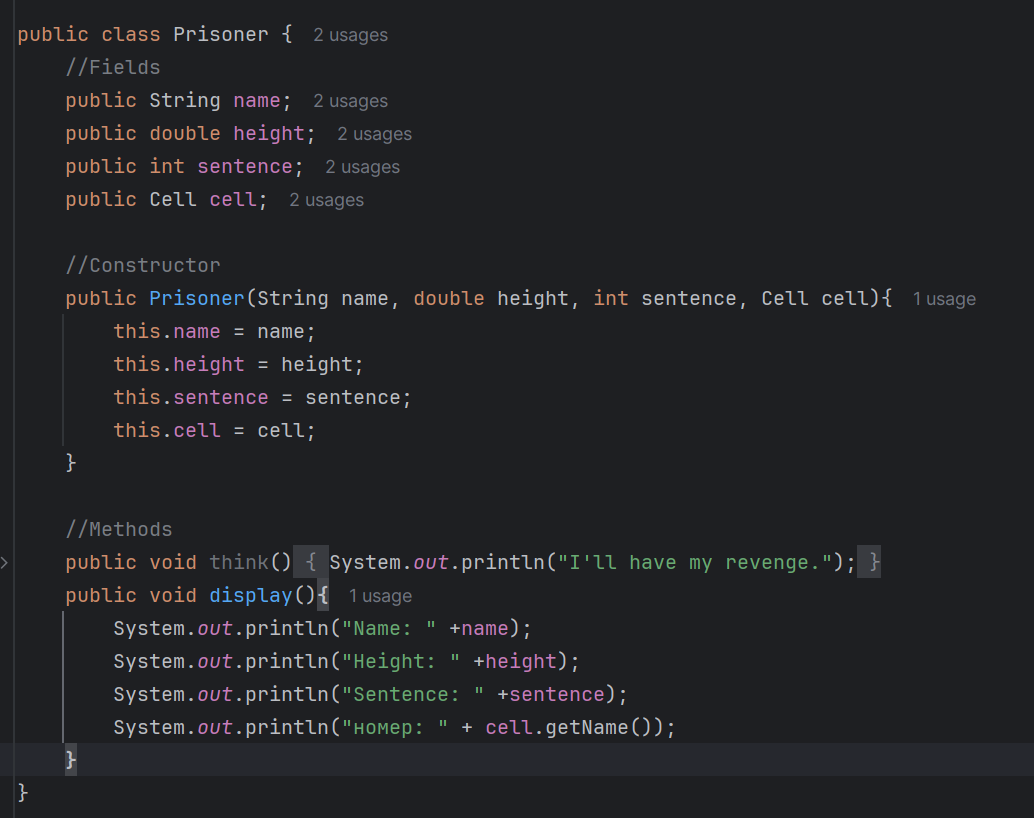
– Заключенный больше не пытается открыть дверь камеры.

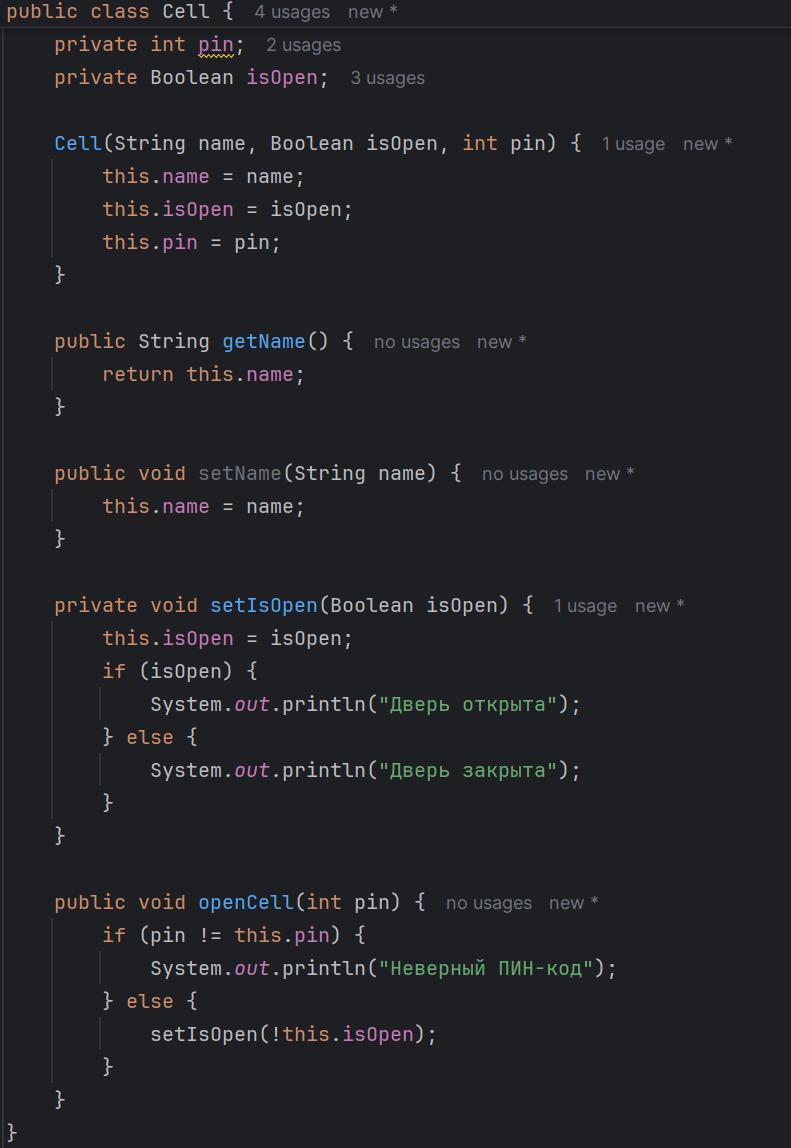
– Выполняется проверка возможности открытия и закрытия двери классом

камеры.

Попробуйте указать правильный и неправильный коды безопасности.

****

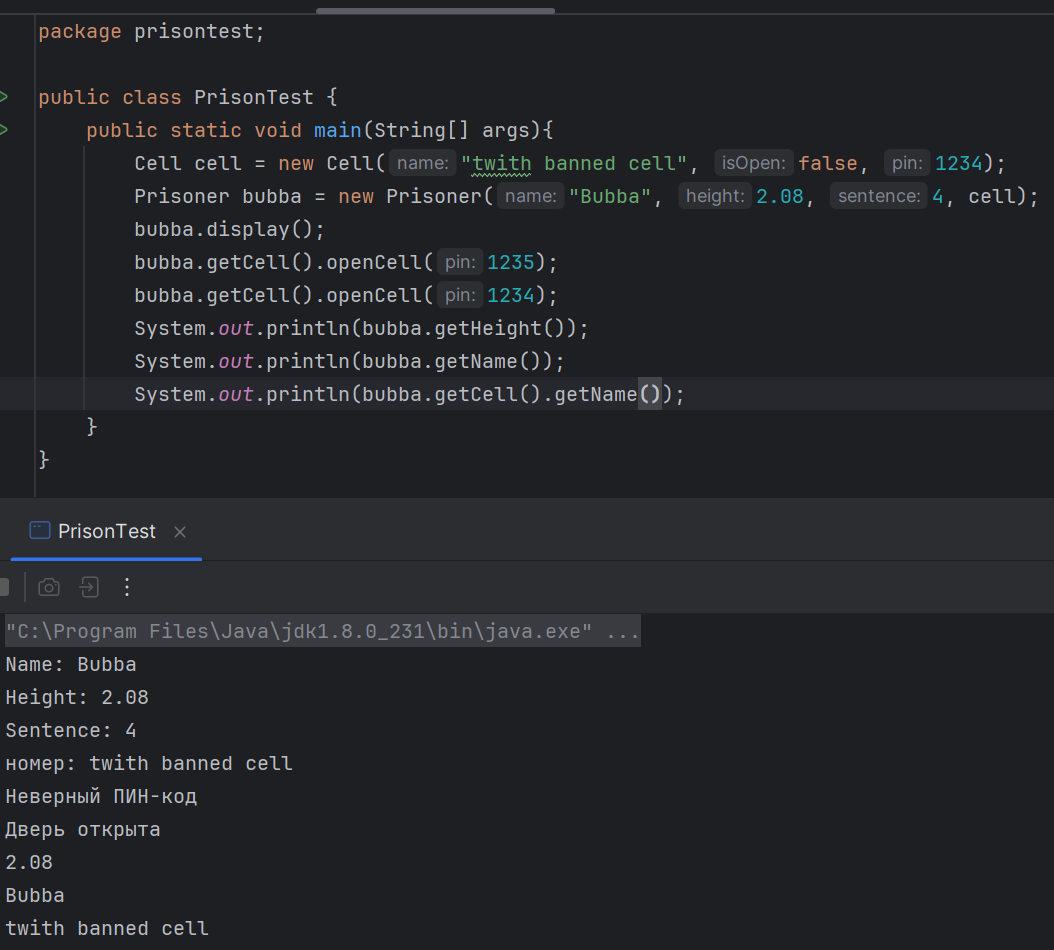
****

****

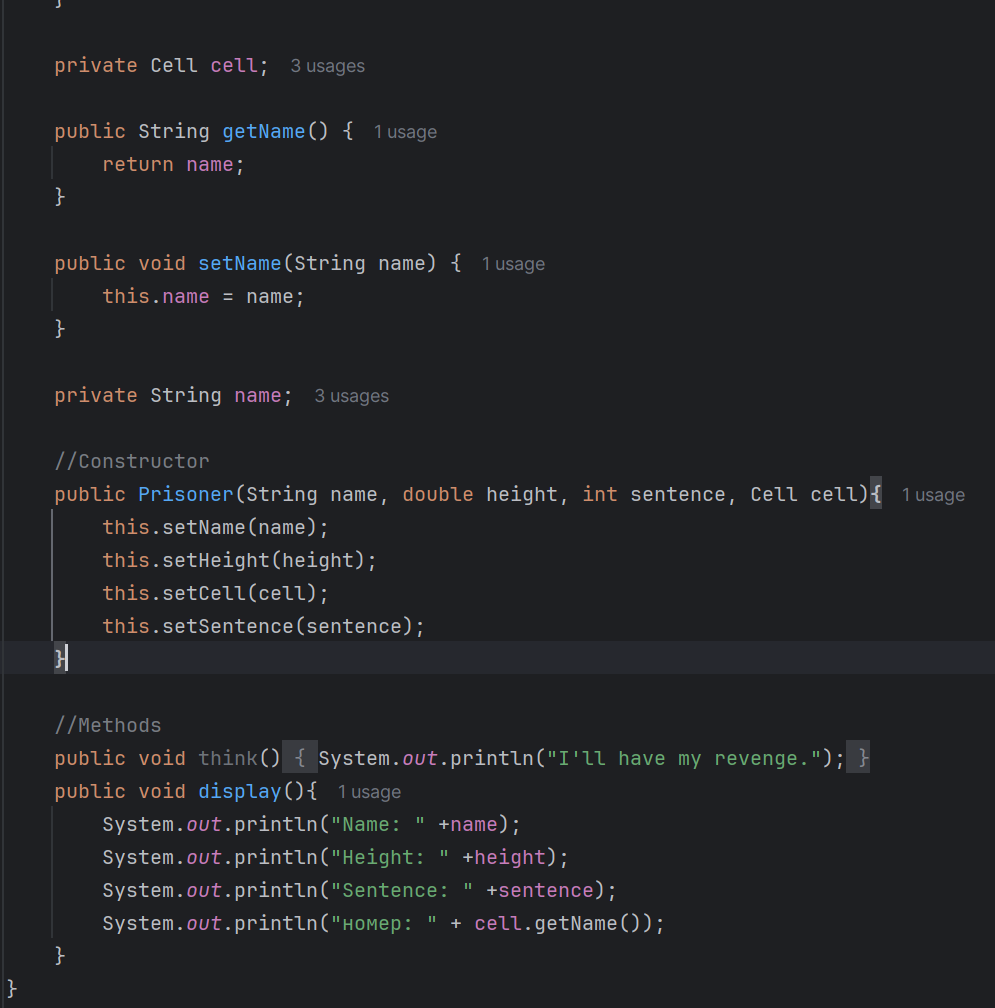
1. **Упражнение 4**

Продолжайте редактировать проект PrisonTest. Выполните инкапсуляцию

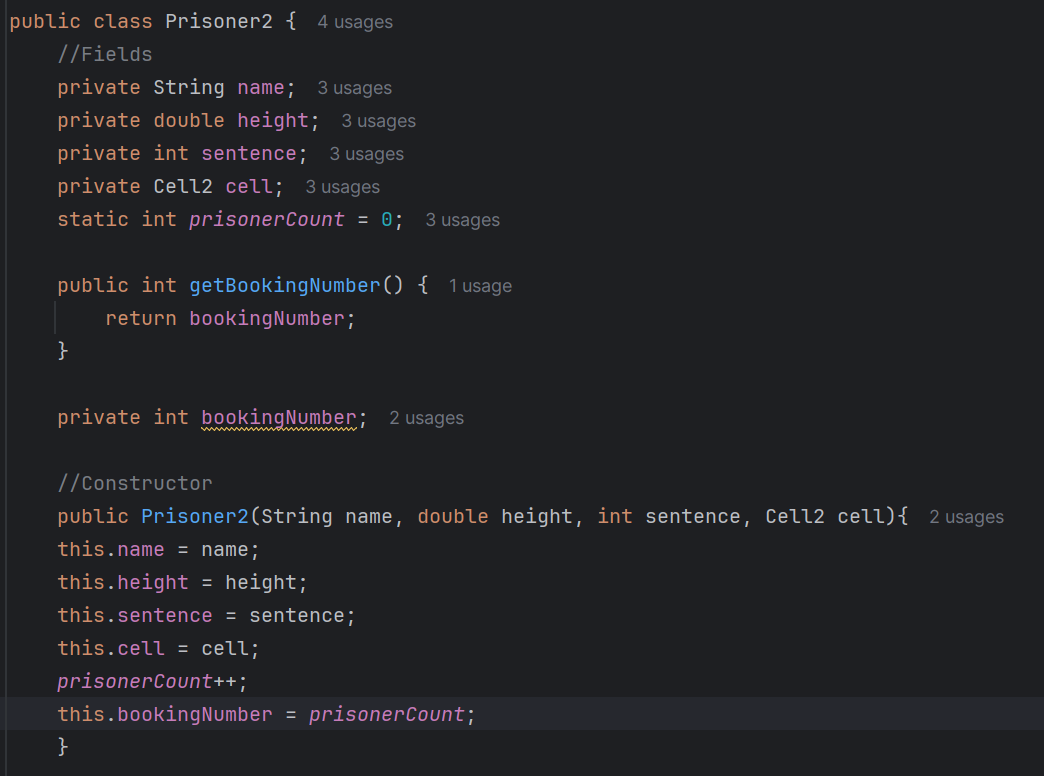
класса Prisoner. Установите для его полей значение private. Укажите методы чтения и устанавливающие методы для каждого поля.

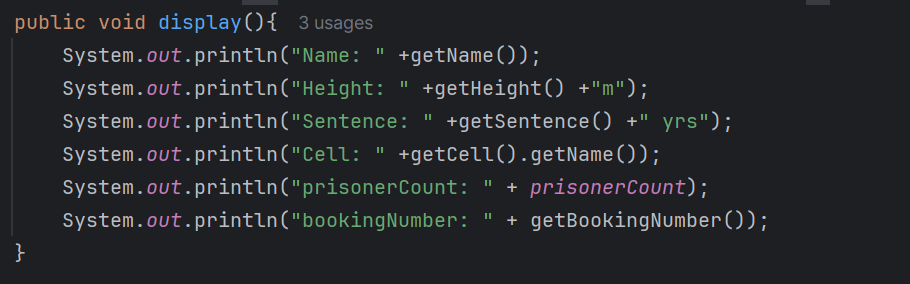


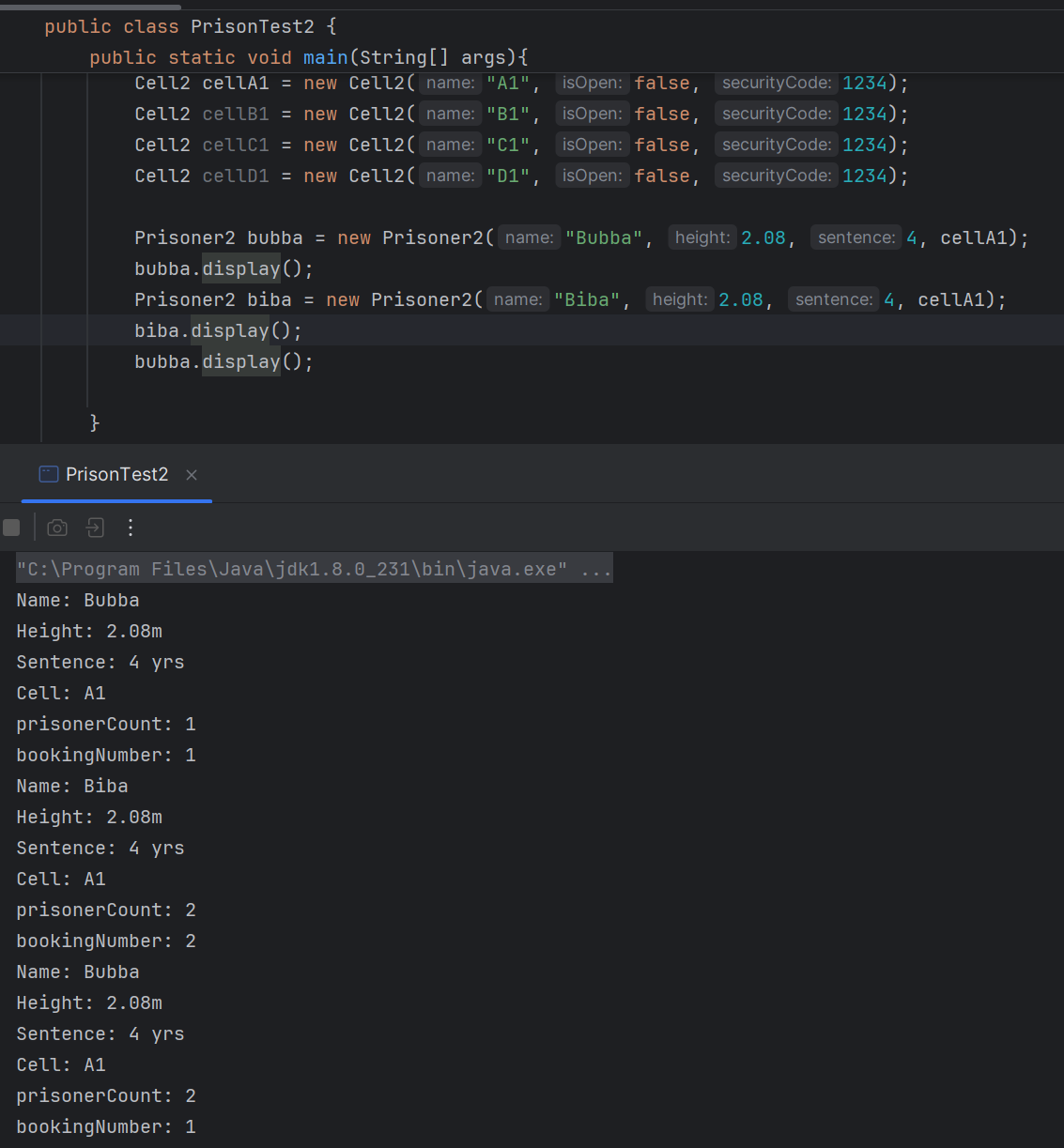




1. **По уроку 7.6**
2. **Упражнение 1**

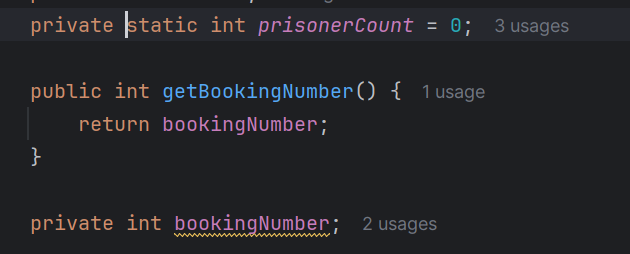
* Продолжайте редактировать проект PrisonTest.
* Измените класс Prisoner:
* Содержи статическое целочисленное поле prisonerCount. Данное поле подсчитывает общее количество заключенных, для которых созданы экземпляры. Выполните инициализацию этого поля до значения 0. Увеличивайте значение поля при каждом создании экземпляра заключенного.
* Включите целочисленное поле bookingNumber. Данное поле было инициализировано с помощью текущего значения prisonerCount.
* Выведите bookingNumber и prisonerCount как часть метода display().
* Создайте экземпляры нескольких заключенных и отобразите сведения о них.

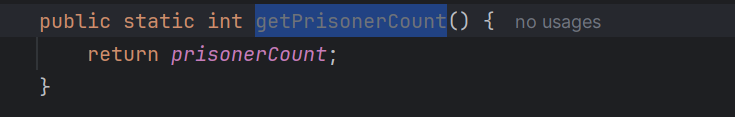


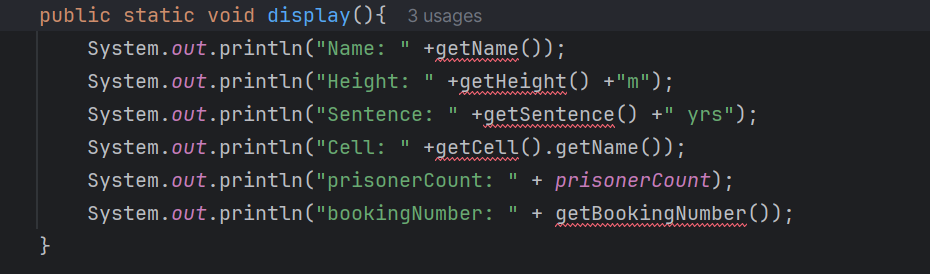


1. **Упражнение3**

* Продолжайте редактировать проект PrisonTest.
* Измените класс Prisoner:
* Выполните инкапсуляцию поля prisonerCount. Сделайте это поле частным и создайте статический метод чтения.
* Попробуйте сделать метод отображения статичным. Что представляют собой неполадки NetBeans?
* В главном методе:
* Вызовите метод чтения, который только что был создан, и выведите возвращенное значение.

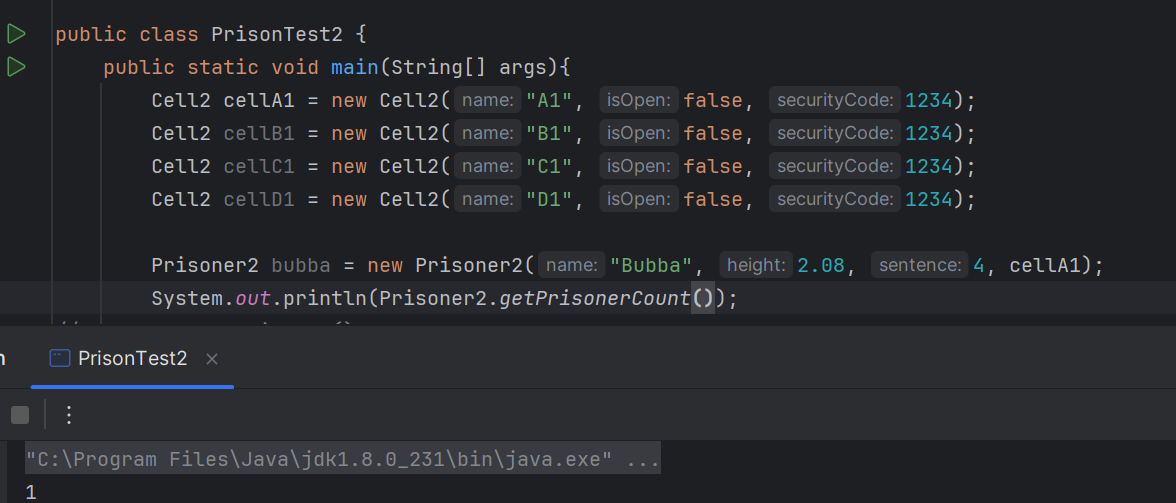






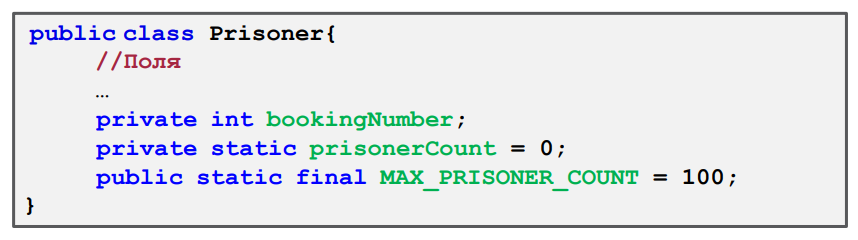
Видим, что возникают ошибки из-за того, что используем нестатические переменные и методы в статичном методе.

Статические поля и статические методы можно вызвать без создания экземпляра объекта. Но переменные экземпляров должны быть связаны с определенным экземпляром. Создается парадокс, если статический метод пытается получить доступ к информации об экземпляре до его создания. Поэтому Java не разрешает, чтобы статические методы содержали переменные или методы экземпляров.



1. **Создание полей static final**

* Вы можете указать для переменных static значение final.
* Помните, что имена переменных final…
* По правилам именования пишутся с заглавных букв.
* Используют нижнее подчеркивание (\_) для разделения слов.



В Java константы должны быть инициализированы при объявлении, вы не можете просто присвоить значение.

* Инкапсуляция не дает манипулировать переменными нежелательным способом.
* Но риска искажения примитивов public static final нет, так как изменить их значения невозможно.
* Это полезно для констант, например π, e, или других значений, которые часто используются в расчетах.
* Эти переменные вызываются непосредственно вместо методов чтения.

1. **Практика 7.**

Написать программу Java, которая моделирует свойства, поведение и взаимодействие объектов в этой галерее игровых автоматов. Вам также потребуется тестовый класс, который содержит главный метод. Используйте главный метод для моделирования определяющих действий программы, таких как создание экземпляров объектов и считывание карт. Все поля должны иметь значение private. Предоставьте метод получения и все необходимые методы установки.

Главный метод:

package src;  
  
import src.exception.InsufficientFundsException;  
import src.exception.InsufficientTicketsException;  
import src.exception.PrizeUnavailableException;  
  
import java.util.List;  
import java.util.Scanner;  
  
public class Test {  
 private static final Terminal *terminal* = new Terminal();  
 private static final Card *card1* = new Card(1, 0, 0);  
 private static final Card *card2* = new Card(2, 0, 0);  
  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
  
 while (true) {  
 System.*out*.println("Выберите действие:");  
 System.*out*.println("1. Загрузить кредиты на карту");  
 System.*out*.println("2. Сыграть в игру");  
 System.*out*.println("3. Перенести баланс кредитов с одной карты на другую");  
 System.*out*.println("4. Перенести билеты с одной карты на другую");  
 System.*out*.println("5. Запросить приз");  
 System.*out*.println("6. Показать баланс карты");  
 System.*out*.println("0. Выход");  
  
 int choice = scanner.nextInt();  
 switch (choice) {  
 case 1:  
 *loadCreditsScenario*(scanner);  
 break;  
 case 2:  
 *playGameScenario*(scanner);  
 break;  
 case 3:  
 *transferCreditsScenario*(scanner);  
 break;  
 case 4:  
 *transferTicketsScenario*(scanner);  
 break;  
 case 5:  
 *claimPrizeScenario*(scanner);  
 break;  
 case 6:  
 *displayBalanceScenario*(scanner);  
 break;  
 case 0:  
 System.*out*.println("Выход из программы.");  
 return;  
 default:  
 System.*out*.println("Неверный выбор. Попробуйте снова.");  
 }  
 }  
 }  
  
 private static void loadCreditsScenario(Scanner scanner) {  
 System.*out*.println("Введите номер карты (1 или 2):");  
 int cardNumber = scanner.nextInt();  
 System.*out*.println("Введите сумму (в рублях) для загрузки кредитов:");  
 int rubles = scanner.nextInt();  
  
 Card selectedCard = (cardNumber == 1) ? *card1* : *card2*;  
 *terminal*.loadCredits(selectedCard, rubles);  
 }  
  
 private static void playGameScenario(Scanner scanner) {  
 System.*out*.println("Введите номер карты (1 или 2):");  
 int cardNumber = scanner.nextInt();  
  
 Game game = new Game(20);  
 Card selectedCard = (cardNumber == 1) ? *card1* : *card2*;  
  
 try {  
 game.playGame(selectedCard);  
 } catch (InsufficientFundsException e) {  
 System.*out*.println(e.getMessage());  
 }  
 }  
  
 private static void transferCreditsScenario(Scanner scanner) {  
 System.*out*.println("Введите номер исходной карты (1 или 2):");  
 int fromCardNumber = scanner.nextInt();  
 System.*out*.println("Введите номер карты назначения (1 или 2):");  
 int toCardNumber = scanner.nextInt();  
 System.*out*.println("Введите сумму кредитов для переноса:");  
 int credits = scanner.nextInt();  
  
 Card sourceCard = (fromCardNumber == 1) ? *card1* : *card2*;  
 Card destinationCard = (toCardNumber == 1) ? *card1* : *card2*;  
  
 *terminal*.transferCredits(sourceCard, destinationCard, credits);  
 }  
  
 private static void transferTicketsScenario(Scanner scanner) {  
 System.*out*.println("Введите номер исходной карты (1 или 2):");  
 int fromCardNumber = scanner.nextInt();  
 System.*out*.println("Введите номер карты назначения (1 или 2):");  
 int toCardNumber = scanner.nextInt();  
 System.*out*.println("Введите количество билетов для переноса:");  
 int tickets = scanner.nextInt();  
  
 Card sourceCard = (fromCardNumber == 1) ? *card1* : *card2*;  
 Card destinationCard = (toCardNumber == 1) ? *card1* : *card2*;  
  
 *terminal*.transferTickets(sourceCard, destinationCard, tickets);  
 }  
  
 private static void claimPrizeScenario(Scanner scanner) {  
 System.*out*.println("Введите номер карты (1 или 2):");  
 int cardNumber = scanner.nextInt();  
 List<PrizeCategory> prizes =*terminal*.getPrizeCategories();  
 for (PrizeCategory prize : prizes) {  
 System.*out*.println("Название приза: " + prize.getPrizeName() +  
 ", Требуемые билеты: " + prize.getTicketsRequired() +  
 ", Доступное количество: " + prize.getPrizesRemaining());  
 }  
  
 System.*out*.println("Введите имя приза:");  
 scanner.nextLine();  
 String prizeName = scanner.nextLine();  
  
 Card selectedCard = (cardNumber == 1) ? *card1* : *card2*;  
  
 try {  
 *terminal*.claimPrize(selectedCard, prizeName);  
 } catch (PrizeUnavailableException | InsufficientTicketsException e) {  
 System.*out*.println(e.getMessage());  
 }  
 }  
  
 private static void displayBalanceScenario(Scanner scanner) {  
 System.*out*.println("Введите номер карты (1 или 2):");  
 int cardNumber = scanner.nextInt();  
  
 Card selectedCard = (cardNumber == 1) ? *card1* : *card2*;  
 *terminal*.displayBalance(selectedCard);  
 }  
  
}

Games:

package src;  
  
import src.exception.InsufficientFundsException;  
  
import java.util.Random;  
  
public class Game {  
 private final int creditsRequired; // Требуемое количество кредитов для игры  
 private final Random random;  
  
 public Game(int creditsRequired) {  
 this.creditsRequired = creditsRequired;  
 this.random = new Random();  
 }  
  
 public void playGame(Card card) throws InsufficientFundsException {  
 // Проверяем, достаточно ли кредитов на карте  
 if (card.getCreditBalance() < creditsRequired) {  
 throw new InsufficientFundsException(card, creditsRequired);  
 }  
  
 payGames(card, creditsRequired);  
  
 int ticketsWon = random.nextInt(10);  
  
 // Обновляем баланс билетов на карте  
 updateTicketBalance(card, ticketsWon);  
  
 // Выводим информацию  
 System.*out*.println("Карточка номер " + card.getCardId() +  
 " выиграла " + ticketsWon + " билетов.");  
 System.*out*.println("Новая итоговая сумма кредитов: " + card.getCreditBalance());  
 System.*out*.println("Новая итоговая сумма билетов: " + card.getTicketBalance());  
 }  
  
 private void payGames(Card card, int creditsRequired) {  
 card.setCreditBalance(card.getCreditBalance() - creditsRequired);  
 }  
  
 private void updateTicketBalance(Card card, int ticketsWon) {  
 card.setCreditBalance(card.getCreditBalance() + ticketsWon);  
 }  
}

Card:

package src;  
  
public class Card {  
  
 public Card(int cardId, int creditBalance, int ticketBalance) {  
 this.cardId = cardId;  
 this.creditBalance = creditBalance;  
 this.ticketBalance = ticketBalance;  
 }  
  
 public int getCardId() {  
 return cardId;  
 }  
  
 public void setCardId(int cardId) {  
 this.cardId = cardId;  
 }  
  
 public int getCreditBalance() {  
 return creditBalance;  
 }  
  
 public void setCreditBalance(int creditBalance) {  
 // не может быть отрицательным  
 this.creditBalance = creditBalance;  
 }  
  
 public int getTicketBalance() {  
 return ticketBalance;  
 }  
  
 public void setTicketBalance(int ticketBalance) {  
 // не может быть отрицательным  
 this.ticketBalance = ticketBalance;  
 }  
  
 private int cardId; // Уникальный идентификационный номер карты  
 private int creditBalance; // Баланс кредитов  
 private int ticketBalance; // Баланс билетов  
}

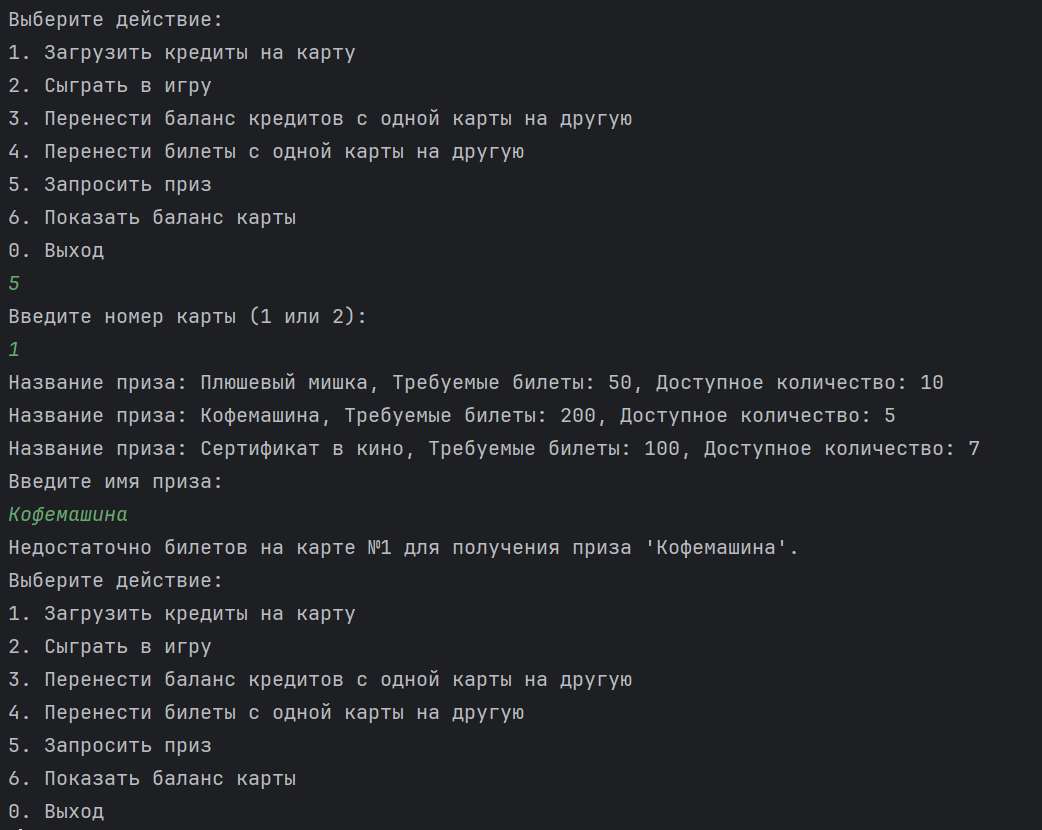
PrizeCategory:

package src;  
  
public class PrizeCategory {  
 private final String prizeName;  
 private final int ticketsRequired;  
 private int prizesRemaining;  
  
 public PrizeCategory(String prizeName, int ticketsRequired, int initialPrizes) {  
 this.prizeName = prizeName;  
 this.ticketsRequired = ticketsRequired;  
 this.prizesRemaining = initialPrizes;  
 }  
  
 public String getPrizeName() {  
 return prizeName;  
 }  
  
 public int getTicketsRequired() {  
 return ticketsRequired;  
 }  
  
 public int getPrizesRemaining() {  
 return prizesRemaining;  
 }  
  
public boolean claimPrize(int ticketsAvailable) {  
 if (prizesRemaining > 0 && ticketsAvailable >= ticketsRequired) {  
 prizesRemaining--;  
 return true;  
 }  
 return false;  
 }  
}

Terminal:

package src;  
  
import src.exception.InsufficientTicketsException;  
import src.exception.PrizeUnavailableException;  
  
import java.util.ArrayList;  
import java.util.List;  
import java.util.Random;  
  
public class Terminal {  
 private final List<PrizeCategory> prizeCategories;  
 private Random random;  
  
 public Terminal() {  
 // Инициализация категорий призов. Можно добавить больше призов  
 this.prizeCategories = new ArrayList<>();  
 prizeCategories.add(new PrizeCategory("Плюшевый мишка", 50, 10));  
 prizeCategories.add(new PrizeCategory("Кофемашина", 200, 5));  
 prizeCategories.add(new PrizeCategory("Сертификат в кино", 100, 7));  
 this.random = new Random();  
 }  
  
 public List<PrizeCategory> getPrizeCategories() {  
 return prizeCategories;  
 }  
  
 public void loadCredits(Card card, int rubles) {  
 int creditsToAdd = rubles \* 2;  
 card.setCreditBalance(card.getCreditBalance() + creditsToAdd);  
 System.*out*.println("На карту №" + card.getCardId() + " добавлено " + creditsToAdd + " кредитов.");  
 displayBalance(card);  
 }  
  
 public void transferCredits(Card sourceCard, Card destinationCard, int creditsToTransfer) {  
 if (sourceCard.getCreditBalance() < creditsToTransfer) {  
 System.*out*.println("Недостаточно кредитов на карте №" + sourceCard.getCardId());  
 return;  
 }  
 sourceCard.setCreditBalance(sourceCard.getCreditBalance() - creditsToTransfer);  
 destinationCard.setCreditBalance(destinationCard.getCreditBalance() + creditsToTransfer);  
 System.*out*.println("Переведено " + creditsToTransfer + " кредитов с карты №" + sourceCard.getCardId() + " на карту №" + destinationCard.getCardId());  
 displayBalance(sourceCard);  
 displayBalance(destinationCard);  
 }  
  
 public void transferTickets(Card sourceCard, Card destinationCard, int ticketsToTransfer) {  
 if (sourceCard.getTicketBalance() < ticketsToTransfer) {  
 System.*out*.println("Недостаточно билетов на карте №" + sourceCard.getCardId());  
 return;  
 }  
 sourceCard.setTicketBalance(sourceCard.getTicketBalance() - ticketsToTransfer);  
 destinationCard.setTicketBalance(destinationCard.getTicketBalance() + ticketsToTransfer);  
 System.*out*.println("Переведено " + ticketsToTransfer + " билетов с карты №" + sourceCard.getCardId() + " на карту №" + destinationCard.getCardId());  
 displayBalance(sourceCard);  
 displayBalance(destinationCard);  
 }  
  
 public void claimPrize(Card card, String prizeName) throws PrizeUnavailableException, InsufficientTicketsException {  
 PrizeCategory prizeCategory = findPrizeCategory(prizeName);  
 if (prizeCategory == null) {  
 throw new PrizeUnavailableException("Приза '" + prizeName + "' нет в терминале.");  
 }  
 if (card.getTicketBalance() < prizeCategory.getTicketsRequired()) {  
 throw new InsufficientTicketsException("Недостаточно билетов на карте №" + card.getCardId() + " для получения приза '" + prizeName + "'.");  
 }  
  
 if (prizeCategory.claimPrize(card.getTicketBalance())) {  
 card.setTicketBalance(card.getTicketBalance() - prizeCategory.getTicketsRequired());  
 System.*out*.println("Приз '" + prizeName + "' получен! Осталось " + prizeCategory.getPrizesRemaining() + " таких призов.");  
 displayBalance(card);  
 } else {  
 throw new PrizeUnavailableException("Приз '" + prizeName + "' закончился.");  
 }  
 }  
  
  
 private PrizeCategory findPrizeCategory(String prizeName) {  
 for (PrizeCategory category : prizeCategories) {  
 if (category.getPrizeName().equalsIgnoreCase(prizeName)) {  
 return category;  
 }  
 }  
 return null;  
 }  
  
 public void displayBalance(Card card) {  
 System.*out*.println("Баланс карты №" + card.getCardId() + ": Кредиты - " + card.getCreditBalance() + ", Билеты - " + card.getTicketBalance());  
 }  
}

Результат:



UML-диаграмма классов:

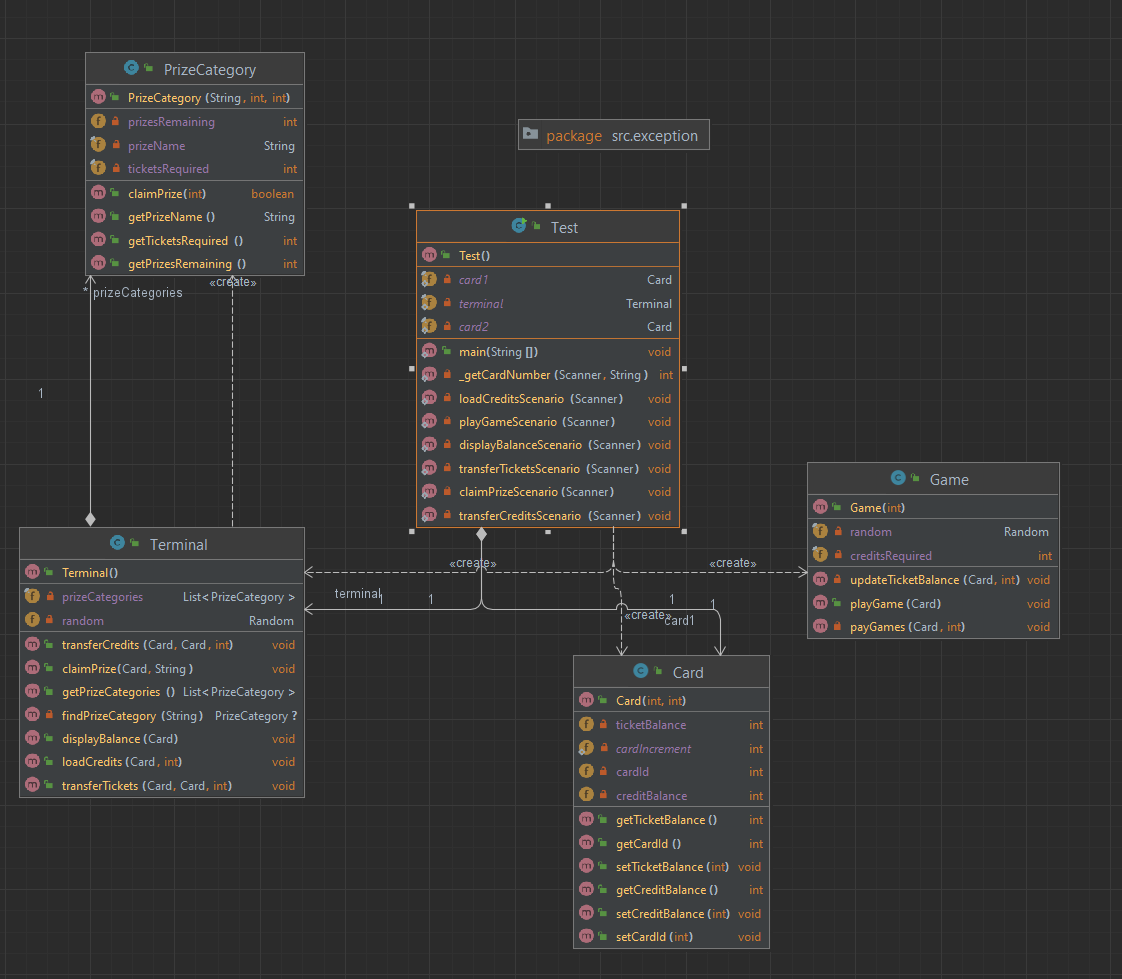
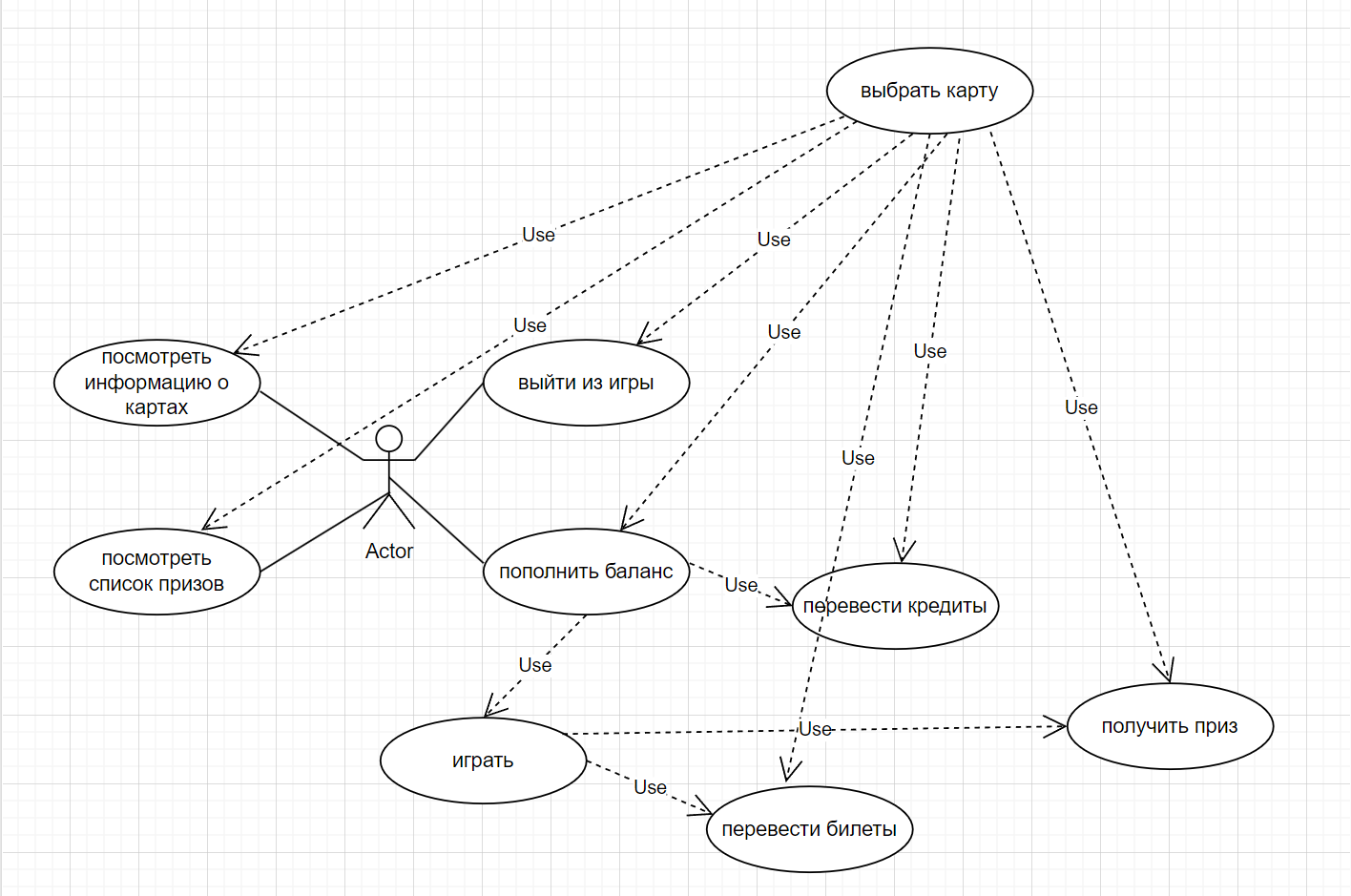
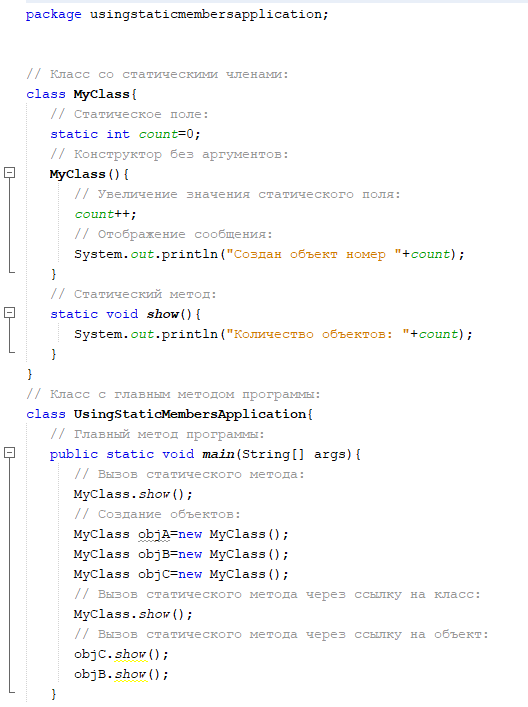


Диаграмма вариантов использования:

****

1. **По книге Васильев, А. Н. Программирование на Java для начинающих**
2. **Листинг 3.5.**

**Программный код проекта UsingStaticMembersApplication**



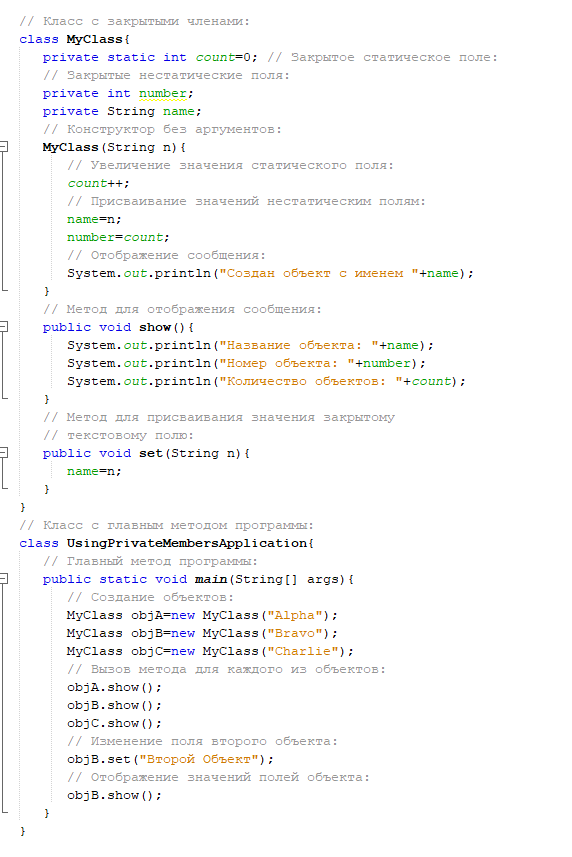
В программе описывается класс MyClass, у которого есть статическое целочисленное поле count, конструктор без аргументов и статический метод show(). Поле count инициализируется с начальным нулевым значением. При вызове статического метода show()в консольном окне отображается сообщение с указанием значения статического поля count. Статическое поле count изменяется при вызове конструктора. В теле конструктора командой count++ на единицу увеличивается значение статического поля, после чего отображается сообщение в окне вывода.

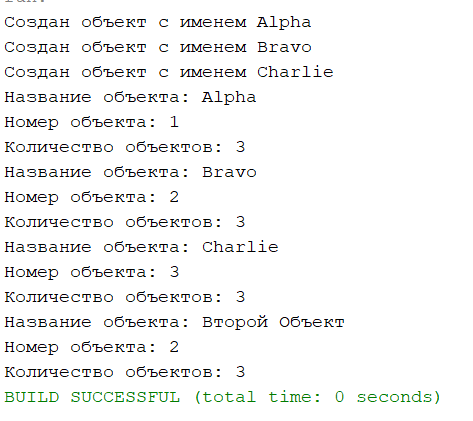
Таким образом, статическое поле count используется для подсчета созданных в программе объектов класса MyClass.

В главном методе программы командой MyClass.show() вызывается статический метод show(). В результате вызова метода в консольном окне появляется сообщение о том, что количество объектов нулевое. На момент вызова метода еще ни один объект не создан. Тем не менее статический метод show(), равно как и статическое поле count, «существуют», и их можно использовать, даже не создавая объект. При обращении к методу show() мы перед именем метода указываем имя класса. Метод show() можно вызвать и «обычным» образом, через ссылку на объект.

1. **Листинг 3.6.**

**Программный код проекта UsingPrivateMembersApplication**





В классе MyClass описано закрытое статическое поле count с начальным нулевым значением, два закрытых нестатических поля (целочисленное number и текстовое name). Также в классе описаны нестатические открытые методы:

* методом show() отображаются значения полей name и number объекта, из которого вызывается метод, а также значение статического поля count;
* метод set() предназначен для присваивания значения закрытому полю name.

Общая ситуация:

* Статическое поле count закрытое и его значение изменяется (увеличивается на единицу) при создании объекта. Других способов (кроме создания объекта) изменить значение поля count нет.
* Закрытое поле number получает значение при создании объекта, после этого нет возможности изменить значение поля.
* Закрытое текстовое поле name получает значение на основе аргумента конструктора. Впоследствии изменить значение поля можно с помощью метода set().