ОТЧЕТ

Ошибка 1 — ошибка границы цикла (off-by-one)

Место: src/services/casino.py, функция run_simulation

Симптом: при заданных n шагах симуляция выполняет n+1 шагов

Как воспроизвести: запустить с –steps 1 и –seed 25

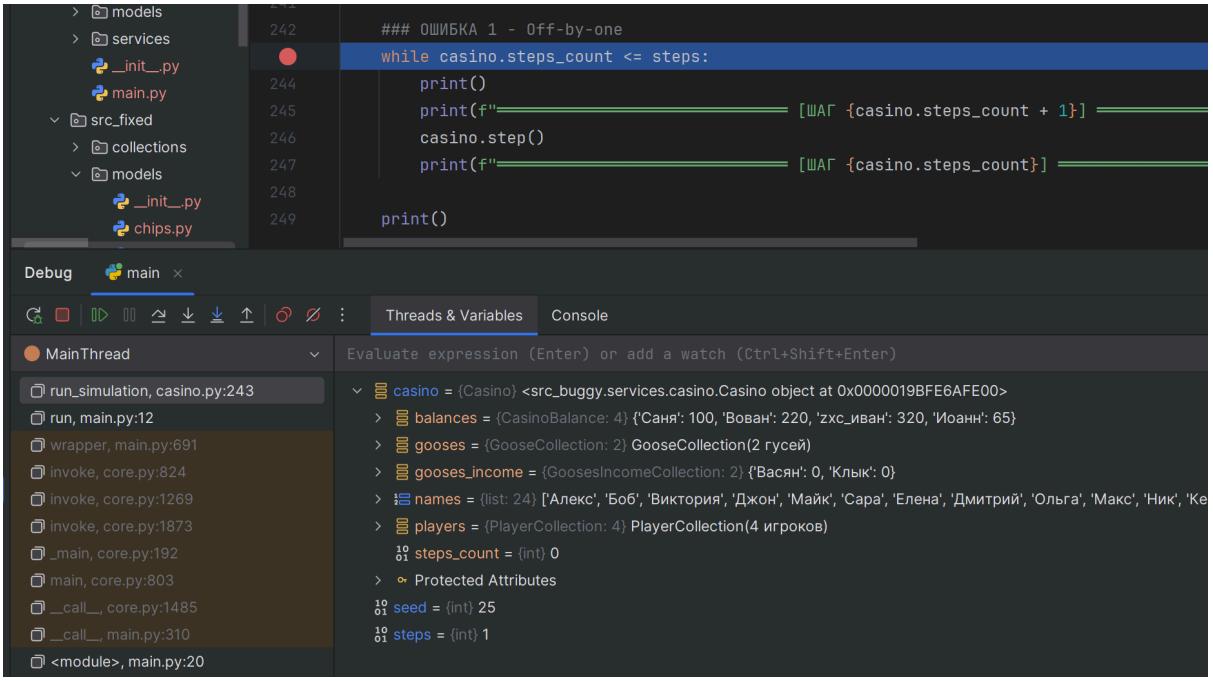
Отладка: установить breakpoint на цикл, который крутит шаги симуляции, и посмотреть как правильно считать шаги

Причина: неверное определена граница цикла

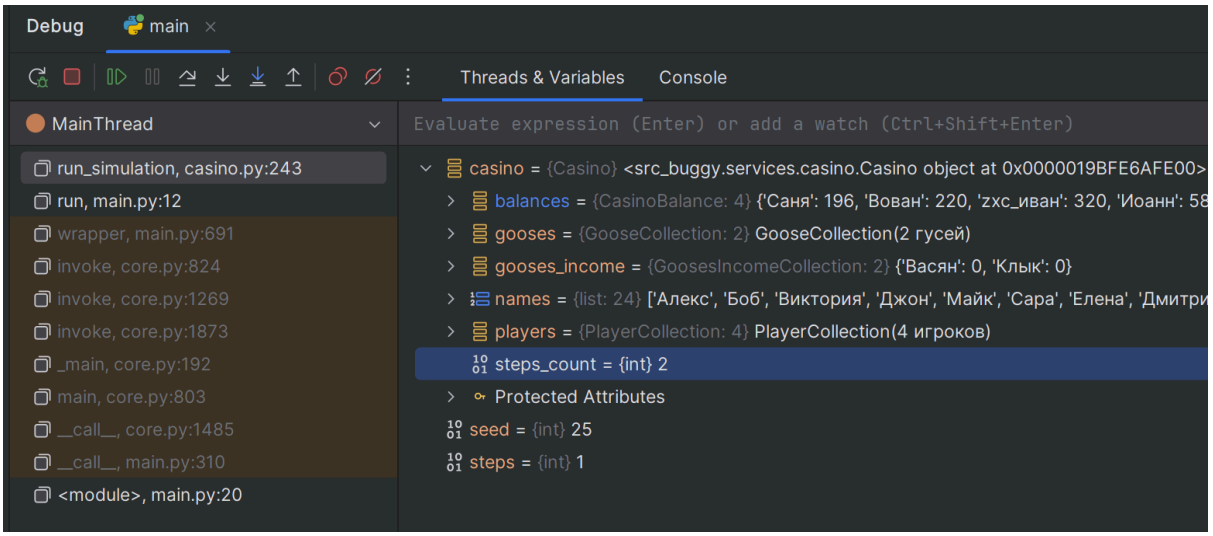
Исправление: в объявлении цикла: **`while casino.steps_count <= steps`** заменить <= на <

Проверка: теперь симуляция выполняет заданное кол-во шагов

Доказательства:
замечаем что в начале атрибут класса Казино steps_count на 1 меньше заданного steps



потом видим, что на последнем шагу steps_count уже на 2 раза увеличился при заданном 1



значит нужно сократить границы цикла

Ошибка 2 — неверное логическое условие

Место: src/services/casino.py, метод panic_action класса Casino

Симптом: если это событие случается, оно правильно выводится в консоль, но под капотом ничего не меняется: игрок не убегает, и рандомный гусь не получает его деньги

Как воспроизвести: запустить симуляцию с –steps 2 и –seed 20

Причина: все, что должно выполняться под капотом, случается при ненулевом балансе игрока, у которого случилась паника, но в условии написано: `if amount < 0` – то есть оно всегда False при ненулевом балансе

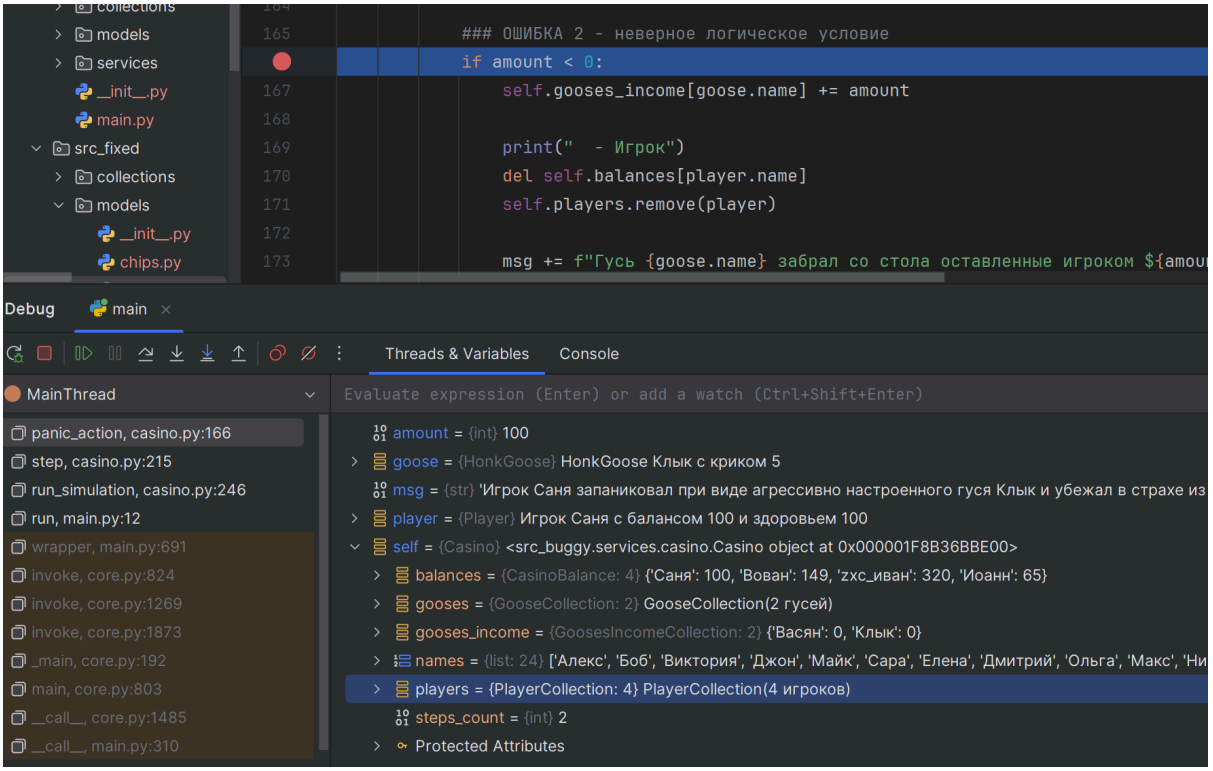
Отладка: поставить брейкпоинты на строчки условия и вызова шагов, сначала дойти до шага с выполнением условия, заметить положительный баланс, а потом следить за коллекциями казино, меняется ли там что-то с этим игроком и доходами гусей

Исправление: замена < на > в условии `if amount < 0`

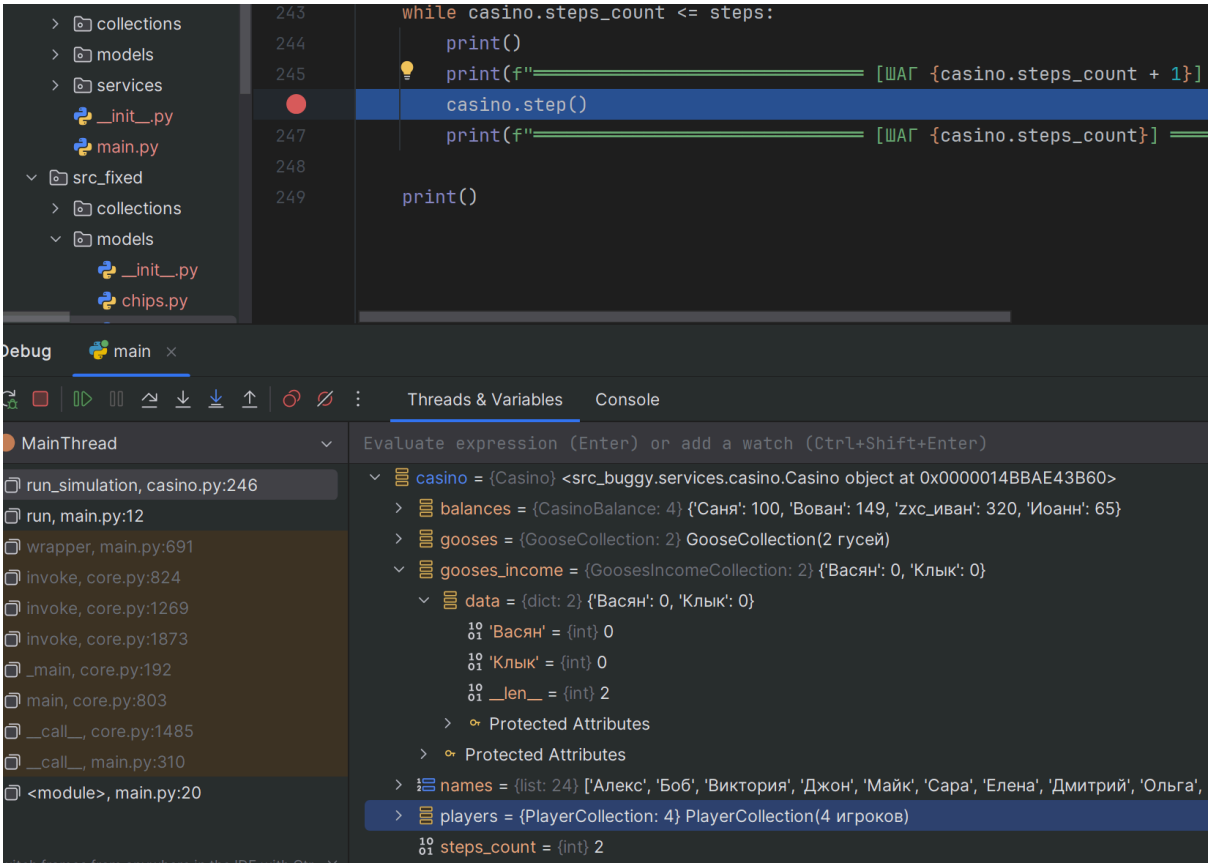
Проверка: Теперь паника пользователя срабатывает корректно под

капотом, пользователь удаляется из всех коллекций казино, а доход рандомно выбранного гуся увеличивается на баланс убежавшего

Доказательства:
замечаем положительный баланс у игрока, который должен будет потом отовсюду удалиться, а его баланс перейти рандомному гусю



а потом заметить, что игрок не удалился, так как число игроков коллекции не изменилось, баланс любого гуся не изменился:



и пойти исправлять условие с amount

Ошибка 3 — сравнение через is вместо ==

Место: src/services/casino.py, метод register_player класса Casino

Симптом: если в симуляции выпадает регистрация игрока, я в сигнатуре не указываю его баланс, поэтому в методе ставится дефолтное значение, а потом если срабатывает условие проверки на дефолтное - баланс выбирается рандомный

Как воспроизвести: запустить с –steps 15 и –seed 20, но сработает оно так же правильно как и с ==

Отладка: поставил брейкпоинт на условие, запустил отладчик

Причина: проверка совпадения с дефолтом: **`if balance is -1`**

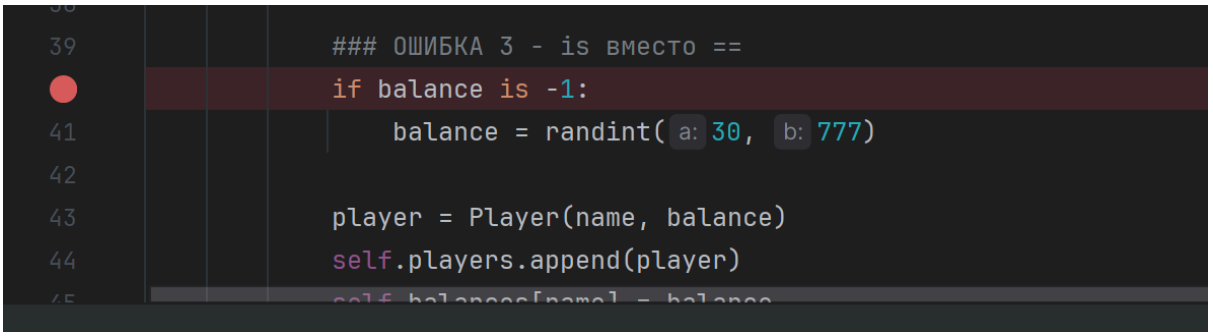
При таком условии проверка строк становится ненадежной потому-что при is значения могут и совпадать, но они могут находится в

разных участках памяти, а is как раз и проверяет, являются ли они одним объектом в памяти

Исправление: заменить is на ==

Проверка: теперь игрок случайный зарегистрируется без проблем, потому что нам главное проверять значения, то есть использовать ==

Доказательства:
точка останова:



я хоть и пытался поймать неправильную работу, все же программа работала правильно, так же как и с ==, но это все же не всегда так возможно. и IDE тоже об этом предупреждает:

Ошибка 4 — перехват слишком общего исключения

Место: src/services/casino.py, метод step класса Casino

Симптом: при возникновении какой-либо ошибки в симуляции программа, когда вызывает случайное событие через try-except, скрывает классифицированную ошибку, ловит просто Exception

Как воспроизвести: запустить с –steps 1 и –seed 22

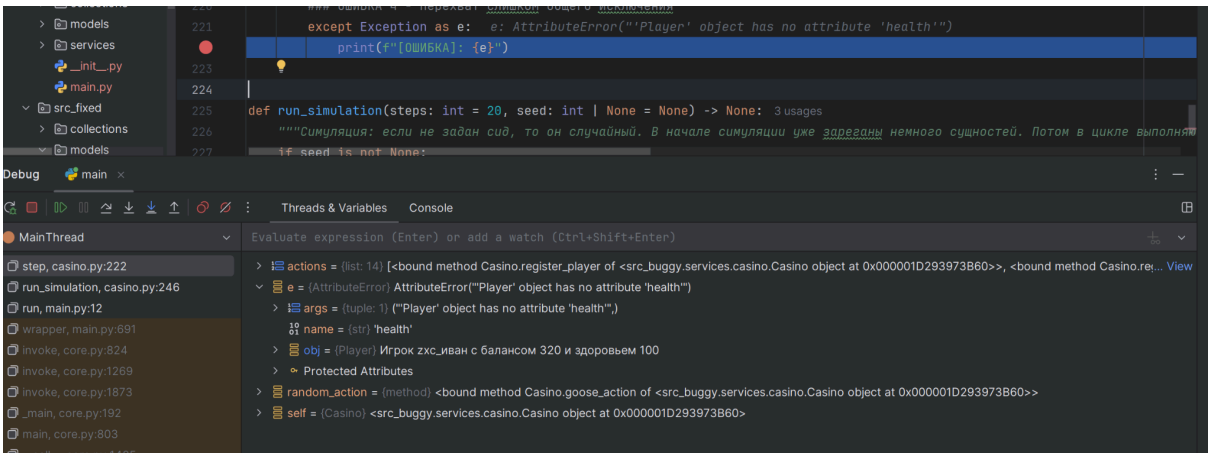
Отладка: breakpoint в строке с принтом внутри **`except Exception as e:`**, поиск настоящих исключений

Причина: **`except Exception as e: print(f"[ОШИБКА]: {e}")`** перехватывает все исключения, которые он охватывает, при этом скрывает информацию об ошибке, необходимую об отладке

Исправление: легче всего улучшить print:
`print(f"[ОШИБКА]{type(e).__name__}: {e}")`

Проверка: теперь при возникновении ошибки, в консоли выводится настоящее исключение на определенном шаге, программа все так же не прерывается

Доказательства: вот случай когда возникает AttributeError (поговорим об этом исключении в следующей ошибке), понимаем, что мы проглатываем классификацию исключения



Ошибка 5 — обращение к несуществующему атрибуту объекта

Место: src/models/gooses.py, метод action класса WarGoose

Симптом: при выполнении действия боевого гуся, он взаимодействует в полем здоровья у игрока, вылетает AttributeError

Как воспроизвести: запустить с –steps 1 и –seed 22

Отладка: установить breakpoint на первую строчку, в которой идет обращение к атрибуту, заметить что существует атрибут здоровья с другим названием

Причина: в методе гуся обращение идет по не существующему атрибуту здоровья объекта игрока: **`victim.health`**

Исправление: указать правильный атрибут здровья везде: **`victim.health -> victim.hp`**

Проверка: теперь взаимодействие происходит с существующим атрибутом, здоровье экземпляра игрока правильно меняется и ошибок не вылетает никаких

Доказательства: