**Абстрактные классы**

[Java Core](https://javarush.ru/quests/QUEST_JAVA_CORE)

[Уровень 2](https://javarush.ru/quests/lectures?quest=QUEST_JAVA_CORE&level=2), Лекция 5

— Привет, Амиго! Новая интересная тема.

— Да сегодня просто день интересных тем!!

— Спасибо!

— Да не за что.

— Помнишь ситуацию, когда мы ввели базовый класс ChessItem для упрощения всех классов шахматных фигур?

— Да.

— Теперь представь, что у каждой фигуры есть метод, который занимается ее отрисовкой на экране. Вызываешь метод, и фигура сама себя рисует в своих текущих координатах. Удобно было бы вынести этот метод в базовый класс?

— Да. После того, что я узнал о полиморфизме, можно было бы вызывать метод отрисовки для всех фигур, независимо от их типа. Примерно так:

Пример

class ChessBoard

{

public void drawAllChessItems()

{

//добавляем фигуры в список

ArrayList items = new ArrayList();

items.add(new King());

items.add(new Queen());

items.add(new Bishop());

//рисуем их независимо от их типа.

for (ChessItem item: items)

{

item.draw();

}

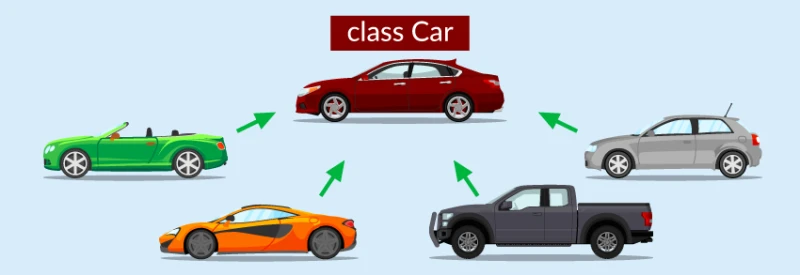
}

}

— Молодец. Именно так. А что бы отрисовал на экране метод draw самого класса ChessItem?

— Не знаю. Такой фигуры ведь в шахматах нет. Значит, и изображения у нее нет.

— Именно. Более того, создавать объекты типа ChessItem – не имеет смысла. Это не фигура из шахмат, всего лишь *абстракция* — класс, который мы сделали для удобства. Так работает абстракция из ООП: мы вынесли важные (общие для всех фигур) данные и методы в базовый класс, а их различия оставили в их классах.

[](https://cdn.javarush.ru/images/article/2129890f-7564-412c-b4b8-e0830036e767/original.jpeg)

Для такого случая в Java есть специальный тип классов – **абстрактные классы**. Вот четыре вещи, которые стоит помнить об абстрактных классах.

**1)** Абстрактный класс может содержать объявление метода без его реализации. Такой метод называется абстрактным.

Пример

public abstract class ChessItem

{

public int x, y; //координаты

private int value; // «ценность» фигуры

public int getValue() //обычный метод, возвращает значение value

{

return value;

}

public abstract void draw(); //абстрактный метод. Реализация отсутствует.

}

3

Задача

Java Core,  2 уровень,  5 лекция

Набираем код Ӏ Java Core: 2 уровень, 5 лекция

Java Core: 2 уровень, 5 лекция. Иногда думать не надо, строчить надо! Как ни парадоксально звучит, порой пальцы «запоминают» лучше, чем сознание. Вот почему во время обучения в секретном центре JavaRush вы иногда встречаете задания на набор кода. Набирая код, вы привыкаете к синтаксису и зарабатываете немного материи. А ещё — боретесь с ленью.

**2)** Абстрактный метод помечается специальным ключевым словом **abstract**.

Если в классе есть хоть один абстрактный метод, класс тоже помечается ключевым словом abstract.

**3)** Создавать объекты абстрактного класса нельзя. Такой код просто не скомпилируется.

|  |  |
| --- | --- |
| **Код на Java** | **Описание** |
| ChessItem item = new ChessItem();  item.draw(); | Этот код не скомпилируется |
| ChessItem item = new Queen();  item.draw(); | А так можно. |

**4)** Если ты наследовал свой класс от абстрактного класса, то нужно переопределить все унаследованные абстрактные методы — написать для них реализацию. Иначе такой класс тоже придется объявить абстрактным. Если в классе есть хотя-бы один нереализованный метод, объявленный прямо в нем или унаследованный от класса-родителя, то класс считается абстрактным.

— А зачем это все нужно? Зачем нужны абстрактные классы? Разве нельзя вместо них использовать обычные? А вместо абстрактных методов просто писать две скобочки в качестве тела метода — {} ?

— Можно. Но эти ограничения сродни модификатору private. Мы специально запрещали с помощью private прямой доступ к данным, чтобы другие программисты и их классы пользовались только написанными нами public-методами.

То же и с абстрактным классом. Тот, кто написал этот класс, не хочет, чтобы создавались его объекты. Наоборот, он рассчитывает на то, чтобы от его абстрактного класса наследовались и переопределяли абстрактные методы.

— Все равно не понятно, зачем усложнять себе жизнь?

— Преимущество этого проявляется в больших проектах. Чем больше классов, тем чётче приходится очерчивать их роли. Ты увидишь преимущество этого, и уже в ближайшем будущем. Все через это проходят.