Наследование. Преимущество наследования

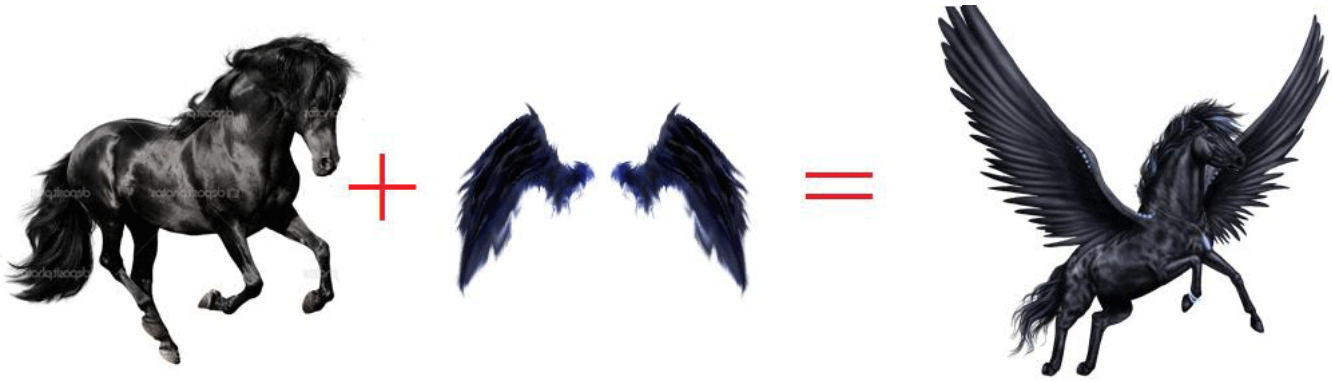
Java Core (/quests/QUEST_JAVA_CORE)
1 уровень (/quests/lectures/?quest=QUEST_JAVA_CORE&level=1), 5 лекция (/quests/lectures/questcore.level01.lecture05)

ОТКРЫТА

— Привет, Амиго! Сейчас будет одна тема, которой, я думаю, ты будешь частенько пользоваться. Это – наследование.

Программирование, для несведущих, неотличимо от магии. Поэтому начну с такой интересной аналогии…

Предположим, что ты волшебник и хочешь создать летающую лошадь. С одной стороны, ты бы мог попробовать наколдовать пегаса. Но т.к. пегасов в природе не существует, это будет очень не просто. Придется очень много делать самому. Куда проще взять лошадь и приколдовать ей крылья.



В программировании такой процесс называется «наследование». Предположим тебе нужно написать очень сложный класс. Писать с нуля долго, потом еще долго все тестировать и искать ошибки. Зачем идти самым сложным путем? Лучше поискать – а нет ли уже такого класса?

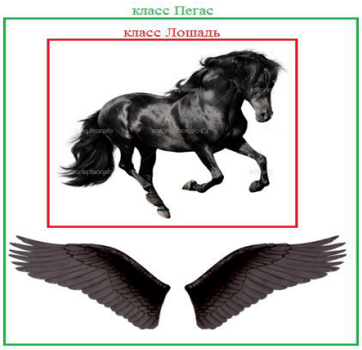
Предположим, ты нашел класс, который своими методами реализует 80% нужной тебе функциональности. Ты можешь просто скопировать его код в свой класс. Но у такого решения есть несколько минусов:

1) Найденный класс уже может быть скомпилирован в байт-код, а доступа к его исходному коду у тебя нет.

2) Исходный код класса есть, но ты работаешь в компании, которую могут засудить на пару миллиардов за использование даже 6 строчек чужого кода. А потом она засудит тебя.

3) Ненужное дублирование большого объема кода. Кроме того, если автор чужого класса найдет в нем ошибку и исправит ее, у тебя эта ошибка останется.

Есть решение потоньше, и без необходимости получать легальный доступ к коду оригинального класса. В Java ты можешь просто объявить тот класс родителем твоего класса. Это будет эквивалентно тому, что ты добавил код того класса в код своего. В твоем классе появятся все данные и все методы класса-родителя. Например, можно делать так: наследуемся от «лошади», добавляем «крылья» — получаем «пегаса»



— Очень интересно, продолжай.

— Наследование можно использовать и для других целей. Допустим, у тебя есть десять классов, которые очень похожи, имеют совпадающие данные и методы. Ты можешь создать специальный базовый класс, вынести эти данные (и работающие с ними методы) в этот базовый класс и объявить те десять классов его наследниками. Т.е. указать в каждом классе, что у него есть класс-родитель – данный базовый класс.

Также как преимущества абстракции раскрываются только рядом с инкапсуляцией, так и преимущества наследования гораздо сильнее при использовании полиморфизма. Но о нем я расскажу завтра. Сегодня же мы рассмотрим несколько примеров использования наследования.

Предположим, мы пишем программу, которая играет в шахматы с пользователем, тогда нам понадобятся классы для фигур. Какие бы ты предложил классы, Амиго?

— Король, Ферзь, Слон, Конь, Ладья и Пешка.

— Отлично. Ничего не упустил.

А какие бы данные ты предложил хранить в этих классах?

— Координаты x и y, а также ее ценность (worth). Ведь некоторые фигуры ценнее других.

— А в чем отличия этих классов?

— Отличия в том, как они ходят фигуры. В поведении.

— Да. Вот как можно было бы описать их в виде классов

```
1  class King
2  {
3  int x;
4  int y;
5  int worth;
6  void kingMove()
7  {
8  //код, решающий,
9  //как пойдет король
10 }
11 }
```

```
1  class Queen
2  {
3  int x;
4  int y;
5  int worth;
6  void queenMove()
7  {
8  //код, решающий,
9  //как пойдет ферзь
10 }
11 }
```

```
1  class Rook
2  {
3  int x;
4  int y;
5  int worth;
6  void rookMove()
7  {
8  //код, решающий,
9  //как пойдет ладья
10 }
11 }
```

```
class Knight
{
int x;
int y;
int worth;
void knightMove()
{
//код, решающий,
//как пойдет конь
}
}
```

```
class Bishop
{
int x;
int y;
int worth;
void bishopMove()
{
//код, решающий,
//как пойдет слон
}
}
```

```
class Pawn
{
int x;
int y;
int worth;
void pawnMove()
{
//код, решающий,
//как пойдет пешка
}
}
```

ЗАДАЧА T Java Core, 1 уровень, 5 лекция ДОСТУПНА

Набираем код

×12

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.

Открыть

— Да, именно так я бы и написал.

— А вот, как можно было бы сократить код с помощью наследования. Мы могли бы вынести одинаковые методы и данные в общий класс. Назовем его ChessItem. Объекты класса ChessItem не имеет смысла создавать, так как ему не соответствует ни одна шахматная фигура, но от него было бы много пользы:

```
1 class King extends ChessItem
2 {
3 void kingMove()
4 {
5 //код, решающий,
6 //как пойдет король
7 }
8 }
```

```
1 class Queen extends ChessItem
2 {
3 void queenMove()
4 {
5 //код, решающий,
6 //как пойдет ферзь
7 }
8 }
```

```
1 class Rook extends ChessItem
2 {
3 void rookMove()
4 {
5 //код, решающий,
6 //как пойдет ладья
7 }
8 }
```

```
1  class ChessItem
2 {
3 int x;
4 int y;
5 int worth;
6 }
7
```

```
1  class Knight extends ChessItem
2  {
3  void knightMove()
4  {
5  //код, решающий,
6  //как пойдет конь
7  }
8  }
```

```
1  class Bishop extends ChessItem
2  {
3  void bishopMove()
4  {
5  //код, решающий,
6  //как пойдет слон
7  }
8  }
```

```
1  class Pawn extends ChessItem
2  {
3  void pawnMove()
4  {
5  //код, решающий,
6  //как пойдет пешка
7  }
8  }
```

×12

ЗАДАЧА T Java Core, 1 уровень, 5 лекция

ДОСТУПНА

★★★★½

Набираем больше кода

Внимание! Объявляется набор кода на JavaRush. Для этого включите режим повышенной внимательности, расслабьте пальцы, читайте код и… набирайте его в соответствующем окошке. Набор кода — вовсе не бесполезное занятие, как может показаться на первый взгляд: благодаря ему новичок привыкает к синтаксису и запоминает его (современные IDE редко дают ему это сделать).

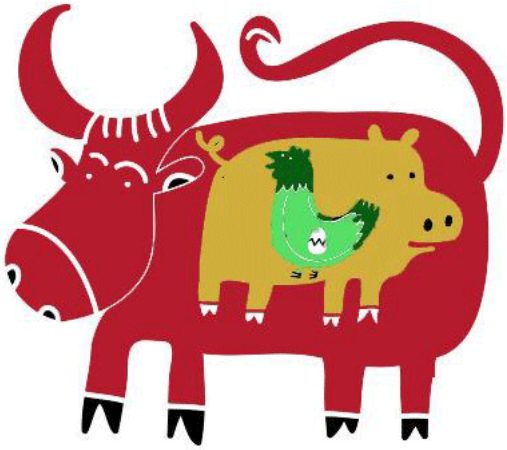
Открыть

— Как интересно.

— Именно! Особенно много преимуществ мы получаем, когда в проекте тысячи различных объектов и сотни классов. Тогда правильно подобранными классами можно не только существенно упростить логику, но и сократить код в десятки раз.

— А что нужно чтобы унаследовать какой-то класс?

— Для этого после объявления нашего класса нужно указать ключевое слово extends и написать имя родительского класса. Унаследоваться можно только от одного класса.



На картинке мы видим «корову», унаследованную от «свиньи». «Свинья» унаследована от «курицы», «курица» от «яйца». Только один родитель! Такое наследование не всегда логично. Но если есть только свинья, а очень нужна корова, программист зачастую не может устоять перед желанием сделать «корову» из «свиньи».

— А если мне хочется унаследоваться от двух классов. Можно же что-то сделать?!

— Почти ничего. Множественного наследования классов в Java нет: класс может иметь только одного класса-родителя. Но есть множественное наследование интерфейсов. Это немного снижает остроту проблемы.

— Ясно. А что такое интерфейс?

— Про интерфейсы я расскажу тебе завтра, а пока давай продолжим разбираться с наследованием.

‹ (/quests/lectures/questcore.level01.lecture04)

×11 › (/quests/lectures/questcore.level01.lecture06)

1 0 0 0 0 +28

Комментарии (24) популярные новые старые

Никита

Введите текст комментария

Юрий Кузнецов 16 уровень 23 марта, 12:54

Хочууу спать xD

Ответить 0

Maxim Sivov 19 уровень, Taraz 16 февраля, 14:59

— Про интерфейсы я расскажу тебе завтра, а пока ложись спать, Амиго!

Ответить 0

Илья 22 уровень, Москва 25 декабря 2017, 11:40

— Очень интересно, продолжай.

Ответить +5

Рустам Сафин 16 уровень, Казань 25 ноября 2017, 17:01

Каков последний рисунок, а.... яйцо, птица, свинья и корова, до боли знакомо))) вот только где сундук с цепями, столетий дуб и общий предок - игла?

Пара вопросов:
1. Возможно ли закольцевание цепи наследования, на примере того же рисунка: яйцо - курица - свинья - корова - яйцо хотя бы для описания репродуктивной функции коровы?

2. А как все же искать класс родитель? Классов море как найти нужный? Есть ли какие-нибудь автоматизированные способы или придётся по старинке - ручками?

Ответить 0

rotarru 20 уровень, Минск 6 января, 08:27

1. Возможно, если вы ответите на вопрос, что было раньше, корова или яйцо.

Ответить +2

Артём 18 уровень 20 ноября 2017, 14:52

Если скопировать к себе методы, то засудят. Если я наследую методы засудят?

Ответить +3

Alex 29 уровень 29 декабря 2017, 08:11

Да ну, это как-то все преувеличено
Докажи еще что код твой
Скажешь я сам придумал и кто что докажет)
Главное чтоб не откровенно прям копи-паст

Ответить +2

Roman Beskrovnyi 33 уровень, Харьков 13 сентября 2017, 13:12

У меня вопрос: почему бы недобавить в ChessItem абстрактный метод move() и переопределить его уже в каждом отдельно? Полиморфизм, епта.

```
1  abstract class ChessItem
2  {
3  int x;
4  int y;
5  int worth;
6
7  abstract void move();
8  }
```

Ответить +11

Andrew Lan 24 уровень 14 сентября 2017, 15:40

+

Ответить 0

Дмитрий 29 уровень — 20 сентября 2017, 15:53

там же было написано, что про полиморфизм позже, Это чтобы не нагружать заранее.

Ответить +2

Юрій Якимчук 16 уровень — 29 сентября 2017, 12:23

А еще лучше определить его в интерфейсе Movable :) Если уж совсем по науке :)

Ответить +1

Агния Старовойтова 25 уровень, Минск — 12 сентября 2017, 13:41

"Есть решение потоньше, и без необходимости получать легальный доступ к коду оригинального класса.
В Java ты можешь просто объявить тот класс родителем твоего класса. "
Так все равно же нужно будет вставлять код родительского класса?
И получается все равно используешь чужой код?
Или я что-то не понимаю?

Ответить +4

Лайт 21 уровень, Санкт-Петербург — 20 сентября 2017, 10:16

Он будет использоваться автоматически, просто методы, реализованные в родительском классе, ты сможешь применять в классах-наследниках. Без переписывания кода.

Ответить +1

Виталий Малый 21 уровень, Днепр — 5 августа 2017, 01:04

наследование это как костыли

Ответить 0

Ольга Козлова 13 уровень, Киев — 26 июля 2017, 01:11

Вот коршуны, без премиум подписки уже не наберёшь халявной материи

Ответить +1

Orion 22 уровень — 4 сентября 2017, 15:39

20$ в месяц это не крыло боинга. Завязывайте уже с этим рабским менталитетом "нам все всё нахаляву должны!".

Ответить +10

Владислав Токарев 12 уровень, Москва — 25 июня 2017, 02:58

Пешка - это не фигура!!!

Ответить 0

Anton Stezhkin 19 уровень — 5 июля 2017, 20:26

а что это?

Ответить 0

Karahanid 40 уровень, Алматы — 14 июля 2017, 13:38

Anton Stezhkin хороший вопрос)

Ответить 0

Orion 22 уровень — 4 сентября 2017, 15:40

пешка - это судьба

Ответить +10

Загрузить еще

ush.ru/) (https://plus.google.com/114772402300089087607) (https://twitter.com/javarush_ru)

JAVARUSH

Программистами не рождаются
© 2018