Лекция 2

Классы как структуры, методы

Классы и объекты

Объекты

- Объекты в реальном мире:
  - Описываются множеством параметров:
    - Длина, ширина, высота, вес, температура и т.д.
    - ФИО, год поступления, специальность и т.д.
  - Состоят из составных частей:
    - Процессор, материнская плана, память
    - Кузов, двигатель, подвеска и т. д.

- Даже такой примитивный объект, как точка на плоскости уже описывается двумя параметрами:
  - Координата X
  - Координата Y

Описание сложных объектов в программе

- Как описывать сложные объекты?
  - Заводить для каждого параметра отдельную переменную?
  - А если объект составной, и каждая часть также составная и т.д. – никаких переменных не хватит.
  - И самое главное, как этим всем управлять (человек способен одновременно оперировать в памяти количество элементов 7 ± 2)?

Как человек думает?

- Рассмотрим задачу:
  - Найти площадь треугольника, заданного тремя вершинами

- Решение:
  - Найти длины сторон треугольника
  - Найти полупериметр
  - Найти площадь, воспользовавшись формулой:

$$S = \sqrt{p(p - a)(p - b)(p - c)}$$

- Обратите внимание, мы в формулировке задачи и в формулировке решения вообще не оперировали координатами точек
  - Конечно, когда нам придется искать длины сторон треугольника по вершинам, нам придется вспомнить о координатах точек, но это будет отдельная задача

Сложность реального мира

- Как правило, программы имеют дело с моделями реального (или виртуального) мира
- Модель – абстрактное представление реальности в какой-либо форме (например, в математической, физической, символической, графической или дескриптивной), предназначенное для представления определённых аспектов этой реальности и позволяющее получить ответы на изучаемые вопросы
- Модель – некое отражение объектов, явлений и т. д. реального мира

Как человек думает – выводы?

- При решении задачи человек оперирует более высокоуровневыми понятиями, объектами и т.д., чем простые переменные (числа, строки и т.п.)

- Поэтому любой современный язык программирования предоставляет средства для описания таких высокоуровневых понятий и объектов

- Предположим вы нашли решение какой-то задачи (математически, придумали алгоритм и т.п.) и вам требуется записать решение в виде программы:
  - Не надо раскладывать ваше решение до отдельных примитивных переменных
  - Надо в программе описать понятия и объекты, которыми вы оперировали в процессе решения (т.е. адаптировать программу под решение, а не наоборот)
    - В этом случае решение программы будет просто изложением хода ваших мыслей

Структуры данных – классы

- Для описания составных объектов в языке Java используются классы
  - На данном этапе изучения программирования мы рассматривать все принципы объектно-ориентированного программирования не будем, а ограничимся только инкапсуляцией (т.е. будем использовать объекты как контейнеры для хранения нескольких переменных)

```
// класс, описывающий точку на плоскости
class Point {
    public double x;
    public double y;
}
```

- Переменные, описанные в классе, называются ***полями класса*** (x, y – поля класса Point)

Структуры данных – классы

```
// класс, описывающий точку на плоскости
class Point {
    public double x;
    public double y;
}
```

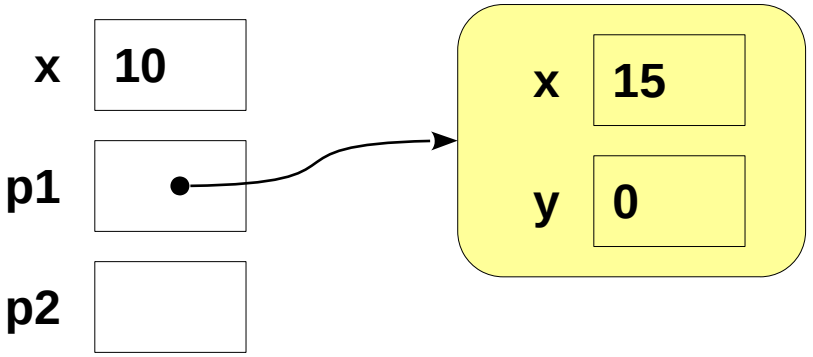
- Теперь у нас появился новый тип составной (сложный) тип данных для описания точек, мы можем объявлять переменные данного типа:

```
...
public class Program {
    public static void main(String[] args) {
        ...
        Point p1;  // p1 – переменная типа Point
        ...
    }
}
```

Работа с переменными типа класс

- Прежде, чем работать с переменной типа Point, нам надо создать **объект** или **экземпляр класса** Point (объект или экземпляр в данном случае синонимы)
- Все дело в том, что в отличие от примитивных типов данных (числа) переменные с типом класс хранят не непосредственно данные, а ссылку на объект, в котором уже находятся данные (объект в данном случае можно понимать как место в памяти, где лежат данные объекта).

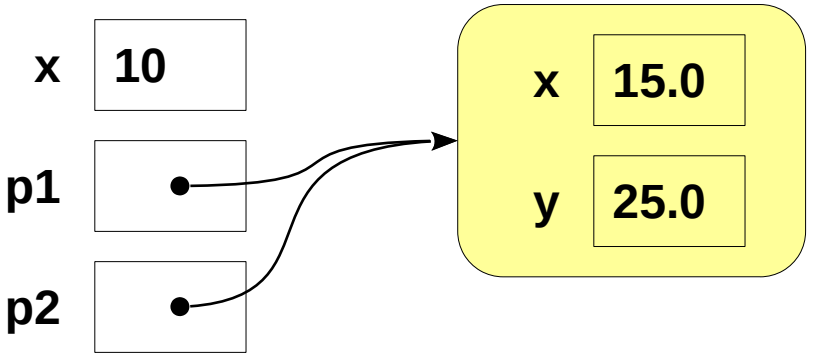
```
int x = 10;
Point p1 = new Point();
Point p2 = null;
p2.x = x + 5;
```



Работа с переменными типа класс

- Обращение к полям объекта происходит через точку
- Присвоение переменной не создает новый объект

```
p2 = p1;
p2.y = 25.0;
```



Конструкторы

- Используются для создания объектов:

```
Point p = new Point(10.3, 0);
```

```
public class Point {
    public double x;
    public double y;

    public Point(double x, double y) {
        this.x = x;
        this.y = y;
    }
}
```

Методы

Запомнить и осознать

- Применение методов (функций) и разработка своих собственных методов должно для вас статить таким же естественным, как ходить
  - Поэтому изучать методы нужно как можно раньше (сразу же с изучением программирования)
  - Не нужно их бояться и противиться (чем грешат многие студенты 1-го курса)

Методы

Статические	Методы объекта (просто методы)
Можно называть функциями (соответствует функциям в процедурных языках)	

Вызывается от класса
 	
`String.format("(%.03f, %.03f)", x, y)`
`Utils.distance(p1, p2)`
	Вызывается от класса
	
	`str.toUpperCase()`
	`p1.distanceTo(p2)`

Все необходимые параметры передаются	Использует как данные из полей объекта, так и передаваемые параметры
Довольно часто описывается в отдельных классах типа Utils	

```
public class Utils {
    public static double distance(
            Point p1, Point p2) {
        double d = Math.sqrt(
                Math.pow(p1.x - p1.x, 2) +
                        Math.pow(p1.y - p2.y, 2)
        );
        return d;
    }
}
```

```
public class Point {
    public double x;
    public double y;

    ...

    public double distanceTo(Point other) {
        return Math.sqrt(
            Math.pow(x - other.x, 2) +
            Math.pow(y - other.y, 2)
        );
    }
}
```

Функции / методы — вопросы терминологии

- Для простоты будем называть ***функций*** статические методы класса (вызываются от имени класса)
  - Что такое класс пока можно себе голову не забивать (для вас это пока просто обязательная синтаксическая конструкция при написании программы)
  - Функции часто называют (особенно в школе) *подпрограммами*
- Обычные методы (вызываются от имени переменной) при этом будем называть просто ***методами***

- Пока вы не изучили основы ООП (объектно-ориентированного программирования), вы будете реализовывать только функции
  - При этом пользоваться уже реализованными вы будете и функциями и методами

Назначение функций и методов

- Функции и методы, как вы могли уже заметить, реализуют некоторую функциональность, которая затем многократно используется вызывается
- Функциональность, которая наиболее востребована в различных проектах уже реализована в стандартной библиотеке языка Java или сторонних библиотеках
  - Реализованной функциональность следует активно пользоваться (принцип «не изобретать велосипед»), но для этого нужно быть с ней знакомым (т.е. читать докумениацию)
- Для своих конкретных задач можно (и нужно) реализовывать свои собственные функции (а в дальнейшем классы и методы)

Как сильно декомпозировать задачу на функции?

- Единого ответа нет — в зависимости от задачи (понимание придет с опытом)
  - Но совершенно точно повторяющийся (похожий) код надо оформлять в виде функций

- Итоговый код программы получился примерно в 2 раза больше, но при этом программа стала проще (каждая функция очень проста, т.е. на любом уровне рассмотрения задача стала более простой)
  - Не всегда функции раздувают код, для больших и сложных задач, наоборот, сокращают
- Мы выделили функции, которые сможем использовать повторно при необходимости (например, в других задачах)

Запомнить и осознать

- Сколь-либо сложные задачи без декомпозиции на более простые задачи (т.е. без разбиения на модули и функции) в принципе не решаются
  - Осознание придет довольно быстро, а пока можете поверить на слово и тренироваться выделять функции на относительно простых задачах

Формальные и фактические параметры функции

- Как можно заметить, о параметрах функции говорят и при описании функции и при вызове функции
  - Параметры, которые объявляются при описании функции называются ***формальными***
  - Параметры, которые передаются функции во время вызова, называются ***фактическими*** (т.е. это конкретные значения, которые передаются функции во время исполнения программы)
    - Соответственно, фактические параметры свои для каждого вызова функции

Формальные и фактические параметры функции

```
static double readDoubleValueFromConsole(String varName) {
    while (true) {
        System.out.printf("Введите %s: ", varName);
        Scanner scanner = new Scanner(System.in);
        String str = scanner.nextLine();
        try {
            return Double.valueOf(str);
        } catch (Exception e) {
            System.out.printf("  -> неверное значение (%s)%n", str);
        }
    }
}
...
double x1 = readDoubleValueFromConsole("X1"),
...
string name = "Высота треугольника";
double x1 = readDoubleValueFromConsole(name);
```

- Здесь формальный параметр функции — varName
- Фактические параметры:
  - В первом вызове – строка "X1"
  - Во втором случае – локальная переменная name (а точнее, значение локальной переменной name)

Формальные и фактические параметры функции

- Формальные и фактические переменные не в коем случае не надо путать (путать здесь в общем-то и нечего)
  - Для этого надо при описании функции полностью абстрагироваться от тех действий / кода, где функция вызывается
  - При описании функции способ взаимодействия с внешним миром для функции — только формальные параметры и возвращаемое значение