ЛЕКЦИЯ № 1

ВВЕДЕНИЕ В ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ И КЛАССЫ

Основные понятия ООП

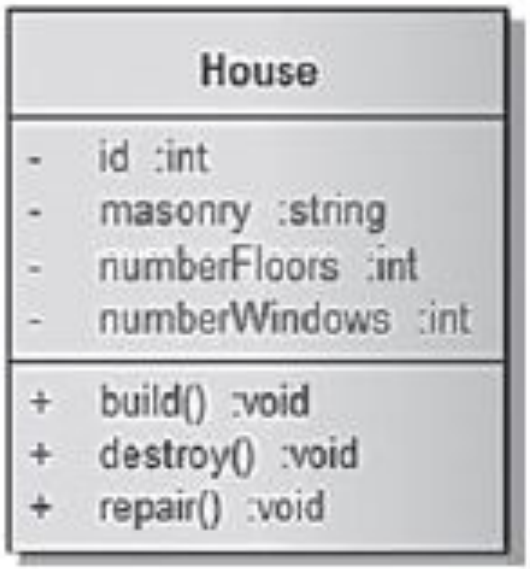
- **Java** является объектно-ориентированным языком программирования .
- **ООП** — методология программирования, основанная на представлении программного продукта в виде совокупности объектов, каждый из которых является **экземпляром конкретного класса.**
- **ООП** использует в качестве базовых элементов взаимодействие объектов.
- **Объект** — именованная модель реальной сущности, обладающая конкретными значениями свойств и проявляющая свое поведение. В применении к объектно-ориентированным языкам программирования понятия объекта и класса конкретизируются.

Основные понятия ООП

- **Объект** – обладающий именем набор данных (полей и свойств объекта), физически находящихся в памяти компьютера, и методов, имеющих доступ к ним.
- **Имя** используется для работы с полями и методами объекта.
- Любой объект относится к определенному классу.
- В классе дается обобщенное описание некоторого набора родственных объектов.
- **Объект** – конкретный экземпляр класса.
- В качестве примера можно привести абстракцию дома или его описание (класс) и реальный дом (экземпляр класса или объект). Объект соответствует логической модели дома, представляющей совокупное описание всех физических объектов.

Графическое изображение класса

- **Класс** принято обозначать в виде прямоугольника, разделенного на три части.
- В верхний прямоугольник помещается имя класса, в средний — набор полей с именами, типами, свойствами класса, и в нижний — список методов, их параметров и возвращаемых значений.
- Реальный объект должен иметь конкретные значения всех полей, например:



id=35, masonry="brick", numberFloors=2, numberWindows=7.

Объектно-ориентированное программирование

основано на принципах:

- инкапсуляции;
- наследования;
- полиморфизма, в частности, «позднего связывания».

Инкапсуляция

- **Инкапсуляция (encapsulation)** — принцип, объединяющий данные и код, манипулирующий этими данными, а также защищающий данные от прямого внешнего доступа и неправильного использования.
- Другими словами, доступ к данным класса возможен только посредством методов этого же класса.

Наследование бывает двух видов:

- **одиночное наследование** — подкласс (производный класс) имеет один и только один суперкласс (предок);
- **множественное наследование** — класс может иметь любое количество предков (в Java запрещено).

Полиморфизм

- **Полиморфизм** (polymorphism) — механизм, использующий одно и то же имя метода для решения похожих, но несколько отличающихся задач в различных объектах при наследовании из одного суперкласса. Целью полиморфизма является использование одного имени при выполнении общих для суперкласса и подклассов действий.
- **Механизм «позднего связывания»** в процессе выполнения программы определяет принадлежность объекта конкретному классу и производит вызов метода, относящегося к классу, объект которого был использован.
- **Механизм «позднего связывания»** позволяет определять версию полиморфного (виртуального) метода во время выполнения программы.
- Другими словами, иногда невозможно на этапе компиляции определить, какая версия переопределенного метода будет вызвана на этапе выполнения программы.

Язык Java

- Объектно-ориентированный язык **Java**, разработанный в компании Sun Microsystems в 1995 году для оживления графики на стороне клиента с помощью апплетов, в настоящее время используется для создания переносимых на различные платформы и опера ционные системы программ.
- **Язык Java** нашел широкое применение в Интернет-приложениях, добавив на статические и клиентские веб-страницы динамическую графику, улучшив интерфейсы и реализовав вычислительные возможности.
- **Но объектно-ориентированная парадигма** и кроссплатформенность привели к тому, что уже буквально через несколько лет после создания язык практически покинул клиентские страницы и перебрался на серверы. На стороне клиента его место заняли языки JavaScript, Adobe Flash и проч.

Системная библиотека

- **Системная библиотека классов языка Java** содержит классы и пакеты, реализующие и расширяющие базовые возможности языка, а также сетевые средства, взаимодействие с базами данных, графические интерфейсы и многое другое.
- Методы классов, включенных в эти библиотеки, вызываются **JVM** (Java Virtual Machine) во время интерпретации программы.

- В Java все объекты программы расположены в динамической памяти — куче данных (heap) и доступны по объектным ссылкам, которые, в свою очередь, хранятся в стеке (stack).
- Это решение исключило непосредственный доступ к памяти, но усложнило работу с элементами массивов и сделало ее менее эффективной по сравнению с программами на C++.
- В свою очередь, в Java предложен усовершенствованный механизм работы с коллекциями, реализующими основные динамические структуры данных.

Объектная ссылка Java

- Необходимо отметить, что **объектная ссылка языка Java** содержат информацию о классе объекта, на который она ссылается, так что объектная ссылка — это не указатель, а дескриптор (описание) объекта.
- Наличие дескрипторов позволяет JVM выполнять проверку совместимости типов на фазе интерпретации кода, генерируя исключение в случае ошибки.
- **В Java изменена концепция организации динамического распределения памяти:** отсутствуют способы программного освобождения динамически выделенной памяти.
- Вместо этого реализована система автоматического освобождения памяти (сборщик мусора), выделенной с помощью оператора new.

Отличие от C++

- **В отличие от C++, Java** не поддерживает множественное наследование, перегрузку операторов, беззнаковые целые, прямое индексирование памяти и, как следствие, указатели.
- В Java существуют конструкторы, но **отсутствуют деструкторы** (применяется автоматическая сборка мусора), не используется оператор **goto** и слово **const**, *хотя они являются зарезервированными словами языка.*

Ключевые и зарезервированные слова языка Java:

abstract	continue	for	new	switch
assert	default	goto*	package	synchronized
boolean	do	if	private	this
break	double	implements	protected	throw
byte	else	import	public	throws
case	enum	instanceof	return	transient
catch	extends	int	short	try
char	final	interface	static	void
class	finally	long	strictfp	volatile
const*	float	native	super	while

- *Кроме ключевых слов в Java существуют **три литерала:** **null, true, false**, не относящиеся к ключевым и зарезервированным словам.*

Пример простого приложения

- Изучение любого языка программирования удобно начинать с программы передачи символьного сообщения на консоль.

```
public class OracleSlogan {
        public static void main(String[ ] args) {
            // вывод строки
            System.out.println("Enabling the Information Age");
        }
}
```

Комментарии:

- Но уже в этом коде заложен фундамент будущих архитектурных ошибок.
- Пусть представленный выше класс является первым из множества классов системы, которые будут созданы в процессе разработки небольшой системы передачи некоторых сообщений, состоящей, например, из двух–трех десятков классов.
- **Строка:**
- *System.out.println("Enabling the Information Age");*
- может встречаться в коде этих классов многократно.

Рассуждения…

- Пусть в процессе тестирования или внедрения системы окажется, что фразу необходимо заменить на другую, например, в конце поставить **восклицательный знак.**
- Во избежание подобных проблем сообщение лучше хранить в отдельном методе или константе (а еще лучше – в файле) и при необходимости вызывать его. Тогда изменение текста сообщения приведет к локальному изменению одной – единственной строки кода.
- В следующем примере **этот код будет переписан** с использованием двух классов, реализованных на основе простейшего применения объектно-ориентированного программирования:

Правильный код:

```
package by.bsu.simple;
public class FirstProgram {
        public static void main(String [ ] args) {
                // объявление и создание объекта firstObject
                SloganAction firstObject = new SloganAction();
                // вызов метода, содержащего вывод строки
                firstObject.printSlogan();
        }
}
```

Описание программы

- Здесь класс ***FirstProgram*** используется для того, чтобы определить метод **main(),** который вызывается автоматически интерпретатором Java и может называться контроллером этого примитивного приложения.
- Метод **main**() получает в качестве параметра аргументы командной строки **String**[]**args**, представляющие массив строк, и является открытым (**public**) членом класса.
- Это означает, что метод main() может быть виден и доступен любому классу. Ключевое слово **static** объявляет методы и переменные класса, используемые при работе с классом в целом, а не только с объектом класса. Символы верхнего и нижнего регистров здесь различаются.
- Тело метода **main**() содержит объявление объекта
- ***SloganAction firstObject = new SloganAction();***
- и вызов его метода
- ***firstObject.printSlogan();***

Описание программы

- Вывод строки «*Enabling the Information Age*» в примере осуществляет метод *println*() (*ln* — переход к новой строке после вывода) статического поля out класса *System*, который подключается к приложению автоматически вместе с пакетом *java.lang*.
- Приведенную программу необходимо поместить в файл *FirstProgram.java* (*расширение.java* обязательно), имя которого должно совпадать с именем *public-класса*.
- Объявление классов предваряет строка *package by.bsu.simple;* указывающая на принадлежность классов пакету с именем *by.bsu.simple,* который является на самом деле каталогом на диске.
- Для приложения, состоящего из двух классов, наличие пакетов не является необходимостью. Однако даже при отсутствии слова *package* классы будут отнесены к пакету по умолчанию (*unnamed*), размещенному в корне проекта.