**LESSON 3**

**Object**

Понятие «объект» не имеет конечного понятия. Можно сказать, что объект – это осязаемая сущность, которая чётко проявляет своё поведение. Если рассмотреть объект в контексте ООП, то можно дать следующее понятие. Объект ООП – это совокупность переменных состояния и связанных с ним методов (операций). Эти методы показывают, как данный объект взаимодействует с окружающей системой.

Java – объектно-ориентируемый язык. Это означает, что конечный продукт будет состоять из объектов.

*Способы создания объектов?*

1. С помощью оператора new:

*Obj object = new Obj();*

1. Используя Class.forName() *:*

*Obj object2 = (Obj) Class.forName("by.epam.Obj").newInstance();*

1. C помощью метода clone() :

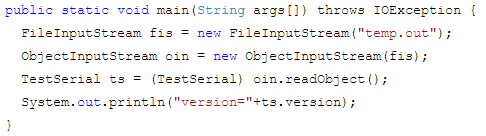
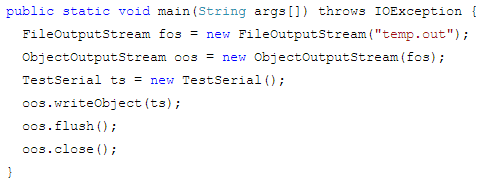
*Obj object = new Obj();*

*Obj object1 = (Obj) object.clone();*

1. C помощью newInstance() :

*Obj object = Obj.class.getClassLoader().loadClass("by.epam. Obj").newInstance();*

1. При десериализации без вызова конструктора



*Механизмы, которые срабатывают при создании объекта?*

1. При создании объекта должен быть загружен байт-код класса (этот процесс размещения байт-кода для данного класса и преобразование этого кода в экземпляр класса Java называется *загрузкой класса*)
2. Выделяется память для хранения свойств объекта
3. Создаётся объектная ссылка, указывающая на эту ячейку памяти
4. Для работы с объектом в будущем ссылка на объект должна быть сохранена как локальная переменная. Иначе, если нет ссылок на данный объект, то объект превращается в «мусор». JVM собирает такие объекты и освобождает память от них. Таким образом, память может быть использована повторно

**Class Object**

*Назначение и использование данного класса?*

Class Object is the root of the class hierarchy. Every class has Object as a superclass. All objects, including arrays, implement the methods of this class.

*Состояние/поведение данного класса?*

Состояние не имеет (нет полей).

Поведение определяется следующими методами (11 публичных методов, 5 обычных и 6 с нативной реализацией):

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| protected native | Object | clone() | throws CloneNotSupportedException | При выполнении метода clone() сначала проверяется, можно ли клонировать исходный объект. Если разработчик хочет сделать объекты своего класса доступными для клонирования через Object.clone(), то он должен реализовать в своем классе интерфейс Cloneable. В этом интерфейсе нет ни одного элемента, он служит лишь признаком для виртуальной машины, что объекты могут быть клонированы. Если проверка не выполняется успешно, метод порождает ошибку *CloneNotSupportedException*.  Если интерфейс Cloneable реализован, то порождается новый объект от того же класса, от которого был создан исходный объект. При этом копирование выполняется на уровне виртуальной машины, никакие конструкторы не вызываются. Затем значения всех полей, объявленных, унаследованных либо объявленных в родительских классах, копируются. Полученный объект возвращается в качестве клона. |
| public | boolean | equals(Object ob) |  | Предназначен для переопределения в подклассах с  выполнением общих соглашений о сравнении содержимого двух объектов; |
| public final native | Class<? extends Object> | getClass() |  | Возвращает объект типа Class, который описывает класс(имя, методы, поля), от которого был порожден этот объект; |
| protected | void | finalize() |  | Вызывается перед уничтожением объекта  автоматическим сборщиком мусора (garbage collection). Отрабатывает автоматически. Если нужно мы можем обратиться напрямую к нему; |
| public native | int | hashCode() |  | Возвращает хэш-код объекта; |
| public | [String](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html) | toString() |  | Возвращает представление объекта в виде строки. Для класса Object и его наследников, не переопределивших toString() |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |