Объектно–ориентированное программирование

Лекция №2

Основы java, объект, класс

к.т.н., Александр Александрович Власов

Вопросы

- объект
- объектная программа
- объектно-ориентированные языки программирования
- сборка мусора
- Точка старта для программы на языке Java
- JVM, JRE, JDK

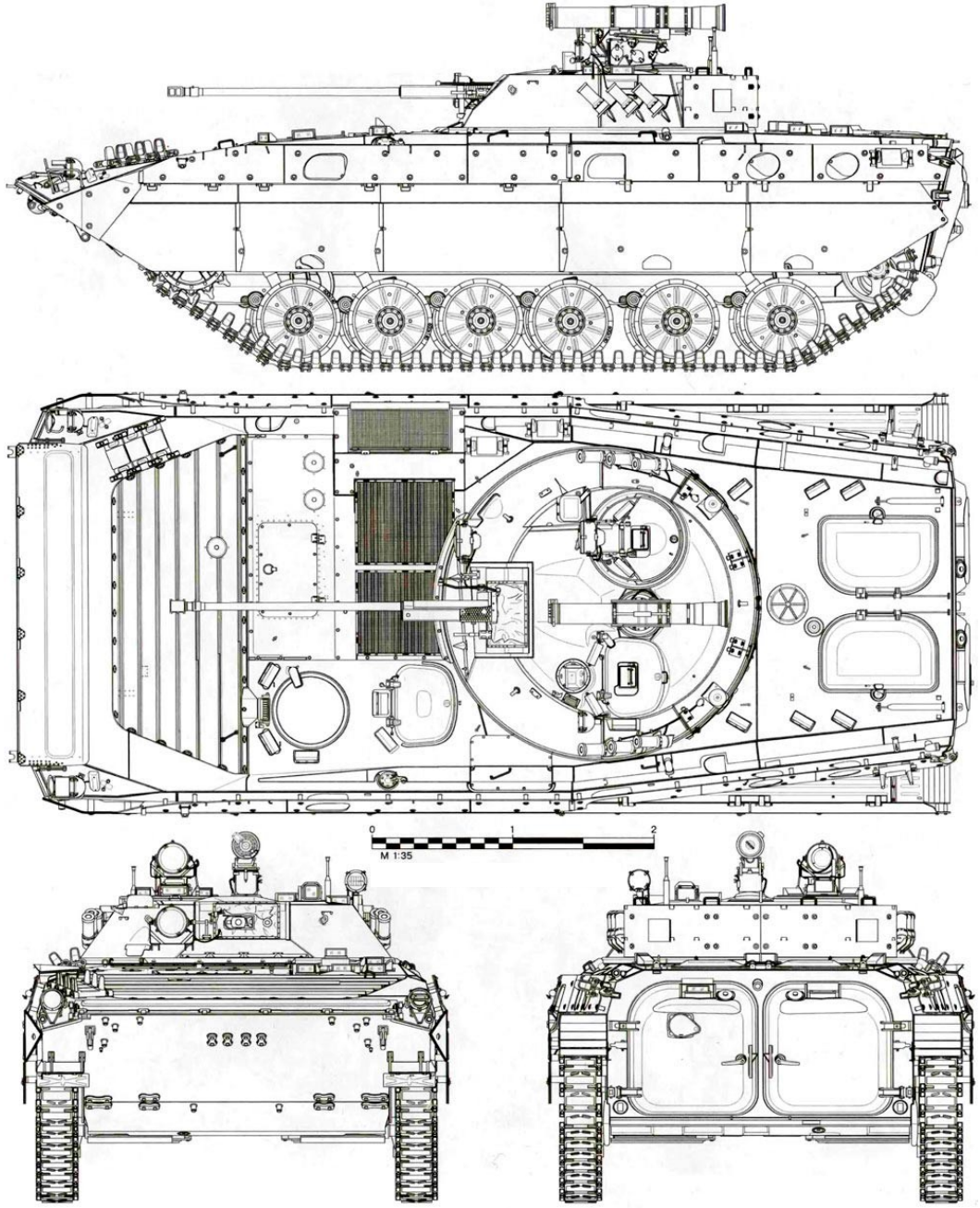
Объект

- Состояние
- Поведение
- Идентичность

- В какой момент появляется?
- Как создать?
- Когда прекратит существование?

Объект

- Объект существует во времени
- Для его создания мы должны знать каким он должен быть
- Необходимо задать нужное начальное состояние
- После уничтожения необходимо освободить занятые ресурсы



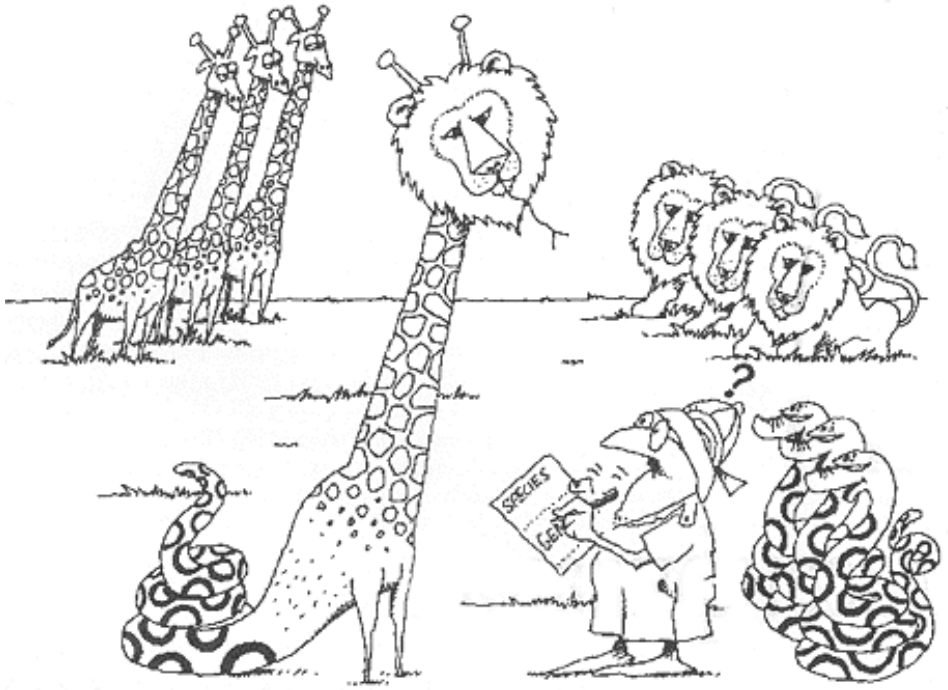
Класс

Нужные для программы объекты можно сгруппировать в классы с общей структурой и поведением

Объект является экземпляром класса

Текст объектно-ориентированной программы состоит из классов

В момент исполнения программы конструируются объекты



Типы данных

01000001 10000000 00000000 00000000

float = 16.0f

int = 1098907648

color = #41800000 

Класс ? Тип данных

Java Language Keywords

- Primitive types: boolean, byte, char, double, float, int, long, short
- Flow control: break, case, continue, default, do, else, for, if, instanceof, return, switch, while
- Access modifiers: private, protected, public
- Package control: import, package
- Error handling: assert, catch, finally, throw, throws, try
- Class, method, variable modifiers: abstract, class, default, extends, final, implements, interface, native, new, static, strictfp, synchronized, transient, var, volatile
- Enumeration: enum
- Others: super, this, void
- Unused (reserved): const, goto
- reserved words: true, false, null

Базовые типы данных

Тип	Константы	Диапазон
boolean	true,false	
byte	-5	8 бит от -128 до 127
short	16000	16 бит от -32768 до 32767
int	-1000000	32 бит от -2147483648 до 2147483647
long	100000000000L	64 бит от -9223372036854775808L до 9223372036854775807L
double	567	64 бит от 4.9e-324 до 1.7e+308, NaN, POSITIVE_INFINITY, NEGATIVE_INFINITY
float	34.9f	32 бит от 1.4e-45f до 3.4e+38f NaN, POSITIVE_INFINITY, NEGATIVE_INFINITY
char	‘a’, 127	UTF-16
ссылка	null	

В языке Java нет беззнаковых типов

Everything in Java binary format files is stored in big-endian order

5

big-endian = 00000000 00000101
little-endian=00000101 00000000

Example - class

```java
package ru.nsu.fit.oop.vlasov;

import java.util.Date;

public class Car {
    private int modelYear;
    private String modelName;
    //public Car(){}
    public Car(int year, String name) {
        modelYear = year;
        modelName = name;
    }
    public Date getModelYear() {
        return new Date(modelYear);
    }
    @Override
    public String toString() {
        return modelName+" "+modelYear;
    }
    public static void main(String[] args) {
        var myCar = new Car( year: 1969,  name: "Mustang");
        var date=myCar.getModelYear();
        var myCars1= new Car[3];
        results[0] = myCar;
        results[1] = new Car( year: 1970, name: "Lada");
        results[2] = new Car( year: 2011, name: "QX56");;
        long results2[] = {results[0], results[1], results[2]};
    }
}
```

- package
- import
- public, private
- new
- default constructor
- constructor
- array
- method
- attribute

Example - stack

```c
// A structure to represent a stack
struct Stack {
    int top;
    unsigned capacity;
    int* array;
};

// function to create a stack of given capacity. It initializes size of
// stack as 0
struct Stack* createStack(unsigned capacity)
{
    struct Stack* stack = (struct Stack*)malloc(sizeof(struct Stack));
    stack->capacity = capacity;
    stack->top = -1;
    stack->array = (int*)malloc(stack->capacity * sizeof(int));
    return stack;
}

// Stack is full when top is equal to the last index
int isFull(struct Stack* stack)
{
    return stack->top == stack->capacity - 1;
}

// Stack is empty when top is equal to -1
int isEmpty(struct Stack* stack)
{
    return stack->top == -1;
}
```

```c
// Function to add an item to stack.  It increases top by 1
void push(struct Stack* stack, int item)
{
    if (isFull(stack))
        return;
    stack->array[++stack->top] = item;
    printf("%d pushed to stack\n", item);
}

// Function to remove an item from stack.  It decreases top by 1
int pop(struct Stack* stack)
{
    if (isEmpty(stack))
        return INT_MIN;
    return stack->array[stack->top--];
}

// Function to return the top from stack without removing it
int peek(struct Stack* stack)
{
    if (isEmpty(stack))
        return INT_MIN;
    return stack->array[stack->top];
}
```

```java
class Stack {
    static final int MAX = 1000;
    int top=-1;
    int a[] = new int[MAX]; // Maximum size of Stack
    boolean push(int x)
    {
        if (top >= (MAX - 1)) {
            return false;
        }
        else {
            a[++top] = x;
            return true;
        }
    }
    int pop()
    {
        if (top < 0) {
            return 0;
        }
        else {
            int x = a[top--];
            return x;
        }
    }
    int peek()
    {
        if (top < 0) {
            return 0;
        }
        else {
            int x = a[top];
            return x;
        }
    }
```

Проект

Исходные коды - *.java

Файлы ресурсов

Файлы системы сборки

Файлы IDE