1. 字节 Byte. 8x二进制位.

Unicode: 0x000000 ~ 0x10FFFF. → 16位 2字节

UTF-16: { U+0000 ~ U+FFFF: 两个字节
U+10000 ~ U+10FFFF: 四个字节

2. jdk: Java Development Kit

jre: Java Runtime Environment

3. 文字变量 { char 必须用单引号
String 必须用双引号.

4.

byte	1B	8bits
short	2B	16bits
int	4B	32bits
long	8B	64bits.
float	4B	32bits
double	8B	64bits

5. 一定要注意静态&非静态!

public static void main() -- 中不允许调用外部非静态变量: public int x;

6. final & static.

	final	static
类属性	✓	✓
方法	✓	✓
类	✓	X
局部变量	✓	X
类中类	✓	✓

final static = static final:

都表示唯一、不变的常量.

7. 创建数组

✓ { int[] ages;
ages = new int[]{19,42};

✓ int[] ages = {19,42};

X int[] ages = new int[]{}; 没有这种构造器.

二维数组要有一个确定维度:

int[][] jag = new int[3][];

✓ { int[] ages;
ages = new int[7];

8. 运算错误.

NaN: 存储的错误 NaN. ← Double.NaN.

整数 / or %0 会抛ArithmeticException.

9. { == : 直接比较(指针地址) <基本类型比较值>
equals: 内容比较.

10. String { String s1 = "M"; → 串池
String s2 = new String(s1); → 内存对象.

s1 != s2.

11. { >> 算术右移 正负不变
>>> 逻辑右移 左侧补零.

运算优先级很低. 仅大于逻辑.

12. cin:

```
import java.util.Scanner;
...
Scanner input = new Scanner(System.in);
int x = input.nextInt();
```

{ while(cin>>x){ } ⇒ { while(input.hasNextInt()){ }

13. switch(x) 要考虑各分支 break;

x的类型: int. byte. short. char. <整数且<4B>

14. 类的修饰符：1. public. abstract. final.

- class A; 包内访问
- public class A;
- abstract class A; 不能实例化
- final class A; 不能被继承.

15. 抽象类 —不一定包含→ 抽象方法

可以没有抽象方法. 有抽象方法一定是抽象类.

抽象类：不能实例化创建对象.

抽象方法：只是声明 无实现(构造器也不行).

√ { 父 抽象类 抽象方法 ↓ 子 抽象类 抽象方法/实现.

√ { 父 抽象类 抽象方法 ↓ 子 非抽象类 必须实现

16. 访问限制.

- public: all. (只要所在类可被访问).
- protected: 包内+子类(继承).
- 缺省: 包内.
- private: 类内.

17. 静态变量:

```
class A{
  static int x = 0;
  ----
}.
```

→ 使用x { √A.x; new A().x; 都可以.

18. 次序

静态初始化块(类初始化块)
↓
非静态初始化块(实例初始化块)
↓
构造器

19. 静态方法只能使用静态属性

20. 静态方法的继承和改写.

```
(父) class cattle{
  ① static void foo(){}
}
class sheep extends cattle
  ② __void foo(){}}
```

左侧编译错误.

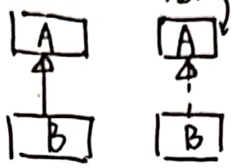
①	②	可行
static	/	×
/	/	√
static	static	√
/	static	×

21. 表示数组的写法

- int []args;
- int ...args;

22. UML图.

继承　接口

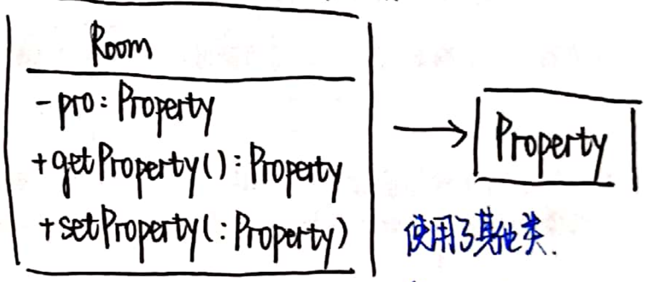


成员: 数组:

-a[1..2]:int ↳ 1..2个成员
⇕
private int[] a;

方法: +foo(a:int):bool
⇕
public bool foo(int a);

在类中用到其他类对象时要用箭头



使用了其他类.

箭头从使用类指向被关联类

```
public class Room{
  private Property pro;
  ⋮
}
```

23. 类的继承. 子类改写父类的方法: 除了运行代码其他要一模一样 (父类private等不可见除外).

```
class A{
  public void foo(){
  }
}
```

```
class B extends A{
  public int foo(){
  }
}
```

报错. 可访问限制可以扩大

24. 父类对象引用子类

a t = new b();

此时 instanceof { t为a √　t为b √

getclass中输出t为b.

编译时	t为a
运行时	t为b

但使用同名静态变量 t.x 时 用的是a的. 因为是在常量池中

25. jdk：编译器(javac)
jre：核心集成API.用户界面.发布技术.虚拟机

26. JDK1.2 { J2ME mobile. / J2SE standard / J2EE enterprise } JDK1.5后 → Java ME / Java SE / Java EE.

公司：Sun —2009→ Oracle

27. Unicode → 16位. 2字节.

28. 非数之间不相等. NaN != NaN.

29. a / b.

a	b	结果
double	0.0	NaN
int	0.0	NaN
double	0	→ NaN
int	0	抛异常 ArithmeticException: / by zero

30. ~~缺省~~构造器不返回值. return不要写!

31. 常用包.
java.lang <自动导入> Object. Throwable. Class. String. Math. System. Thread.
java.util Arrays. List. Set
java.io 输入输出相关.

32. Integer { -128 ≤ x ≤ 127 内存中同一元素 / others. 新建Integer对象实例. }
与String
String { 有new → new String(" ") 新建对象. / 无new → String s = " "; 串池中元素 }

36. try {
可能出错的语句　　try块执行多次 捕获多次异常
} catch (Exception e) {　← 可以有多个catch 来捕获不同类型的异常
错误的处理方法
} finally ← 总被执行

37. Throwable { Exception { SQL Exception / IO Exception / Runtime Exception { IndexOutOfBoundsExce- / NullPointerE- / ArithmeticE- / ClassCastE- } } / Error }

38. try...catch 语句与 throws 二选一.
throws：自己不处理异常. 由JVM虚拟机打印信息结束.
↳ 方法签名.
public static void main (String[] args) throws IOException
且子类抛出异常 ≤ 父类抛出异常.
eg: 父类方法 throws IOException
子类方法 throws Exception　⊗ IOE... < E...

39. 调用会抛异常的方法(如main调用ok).
{ ...ok throws Exception
...main (String[] args) {
try...catch...
} }
{ ...ok throws Exception
...main (String[] args)
throws Exception
} }

40. 数组成份的数目是在使用new创建时确定的，而不是声明时确定的.

声明	创建
int[] a1 =	new int[3];
int[] a2 =	new int[]{1,2,3};

41. 包装类
Type.valueof() ⇒ 相当于构造器 Type(--);
Type.parseType(" ") ⇒ 字符串→类型. eg: "42" → 42(int)

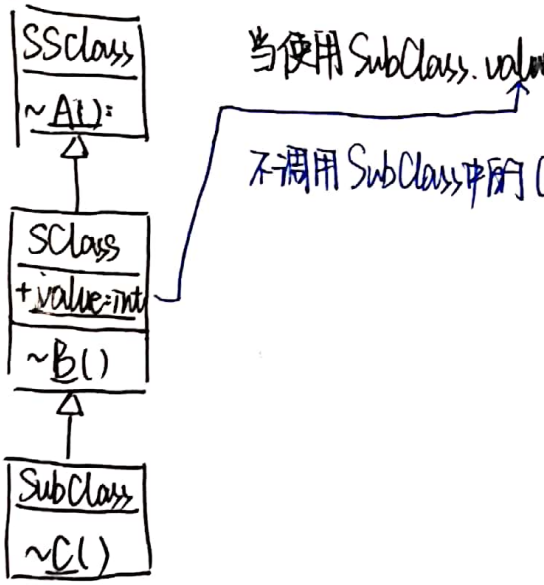
42. 构造器中：
class Hello {　　构造器不可调用自身
public Hello() { };
public Hello(int x) {
this(); 或子类构造器中的super(); ⇒ 必须在第一行.
}

43.

	方法重载/复用	方法改写/重写 → +static
名称	相同	相同
返回值	无要求	(或子类)相同
参数	不同	相同
访问控制	无要求	子类 ≥ 父类

final/static方法不能改写

46. 通过子类引用父类的静态字段，不会导致子类初始化：



47. 类初始化 如果引用的变量是常量 final static 不用初始化类和实例. ←常量存在常量池中.

48. 数组. 类的数组不会初始化类.

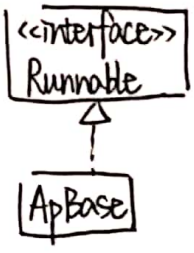
49. 抽象类构造函数通常用 protected. 而且只能通过子类使用

50. 抽象类实例化 { 通过子类创建然后使用子类属性. 引用子类：父 fc = new 子();

51.

	抽象类	接口
属性变量	✓	X
方法(非抽)	✓	X
抽象方法	✓	✓
常量	✓	✓
构造器	✓	X

52. 接口本身不能实例化，但可以用new创建数组




```
Runnable[] r = new Runnable[10];
for (int ... ){
    r[i] = new ApBase();
}.
```

53. 例：

```
class father{                                   但父类没有无参super();
    public father(String s){System.out...}
}
public class child extends father{ → 会自动调用
    public child(String s){System.out...} super();
    public static void main(String[] args){
        new child("c"); //编译错误.
    }
```

54. Object.equals() vs Object.hashCode():

hashCode()相等] → equals()相等
~~也证等其他相等~~

改写equals也应改写hashCode.

55. 线程.

(1) `A extends Thread {`
改写run() ←每个线程的内容. `A t1 = new A();`
主函数创建线程对象并用 t1.start();

(2) `A implements Runnable {...`
改写 run();.
主函数 `Thread t1 = new Thread(new A());`
t1.start();

56. 构造器不是方法. 不可以被子类继承.

57. JAVA_HOME：jdk安装目录.
Path: ...jre\bin <完整jdk的bin目录>
可在任何地方使用javac, java命令
CLASSPATH: .;%JAVA_HOME%lib;...
搜索class时先搜当前目录的class.