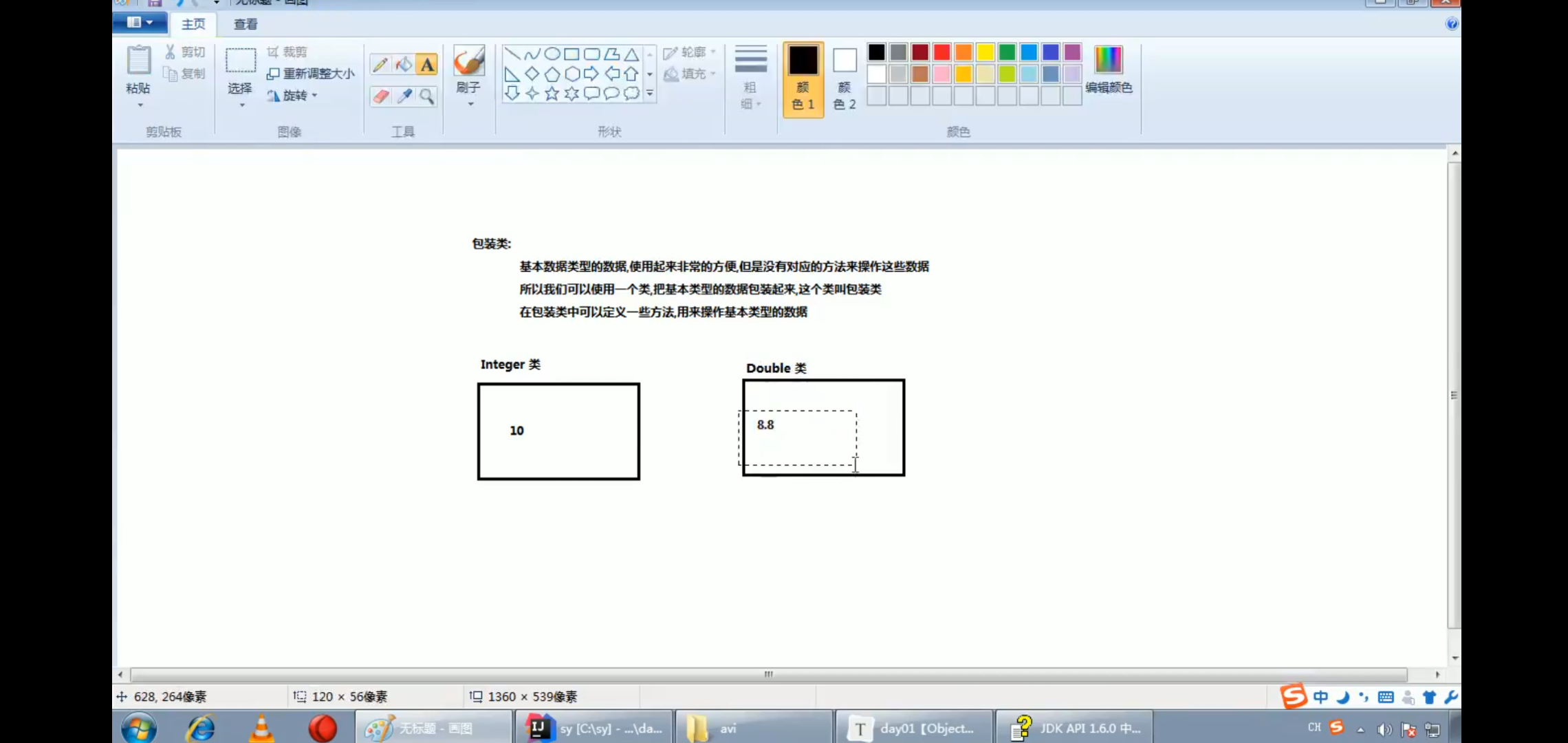
1. 包装类

5.1 概述



基本类型 对用的包装类（位于Java.long包中）

byte Byte

short Short

int Integer

long Long

float Flaot

double Double

char Character

boolean Boolean

3.2 装箱与拆箱

基本类型与对应的包装类之间，来回转换的过程称为”装箱”与”拆箱”:

装箱：从基本类型转换为对应的包装类对象。

拆箱：从包装类对象转换为对应的基本类型。

*/\*  
装箱：把基本类型的数据，包装到包装类中（基本类型的数据 -> 包装类）  
构造方法：  
 Integer(int value) ：构造一个新分配的 Integer 对象，它表示指定的 int 值。  
 Integer(String s) ：构造一个新分配的 Integer 对象，它表示 String 参数所指示的 int 值。  
 注意：传递的字符串必须是基本数据类型的字符串，否则会抛出异常。"100"，正确 "a" 抛出异常  
静态方法：  
 static Integer valueOf(int i):返回一个表示指定的 int 值的Integer实例。  
 static Integer valueOf(String s):返回保存指定的 String 的值的 Integer 对象。  
拆箱：在包装类中取出基本类型的数据（包装类 -> 基本数据类型）  
成员方法：  
 int intValue() 以int类型返回该Integer的值。  
 \*/***public class** Demo05Integer {  
 **public static void** main(String[] args) {  
 *// 装箱：把基本类型的数据，包装到包装类中（基本类型的数据 ->包装类）  
 // 构造方法* Integer in1 = **new** Integer(1); *// 方法上有横线，说明方法已经过时了* System.***out***.println(in1); *// 1 此时1是一个Integer类 ，重写了toString方法* Integer in2 = **new** Integer(**"1"**);  
 System.***out***.println(in2); *// 1  
  
 // 静态方法* Integer in3 = Integer.*valueOf*(1);  
 System.***out***.println(in3); *// 1 此时1为一个Integer包装类* Integer in4 = Integer.*valueOf*(**"1"**);  
 System.***out***.println(in4); *// 1 此时1为一个Character包装类* Integer in5 = Integer.*valueOf*(**"a"**);  
 System.***out***.println(in5); *// NumberFormatException数字格式化异常，不能把 a 变成一个整数。  
   
 // 拆箱：在包装类中取出基本类型的数据（包装类 -> 基本数据类型）* **int** i = in1.intValue();  
 System.***out***.println(i); *// 1 但是此时的 1 代表为int值，而非包装类。* }  
}

5.3 自动装箱与自动拆箱

原因：由于我们经常要做基本类型与包装类之间的转换，从java5（JDK 1.5）开始，基本类类型与包装类的装箱、拆箱动作可以自动完成。

**import** java.util.ArrayList;  
*/\*  
自动装箱与自动拆箱：  
 基本类型的数据和包装类之间可以自动的相互转换。  
 JDK1.5 之后出现的新特征。  
 \*/***public class** Demo05Integer01 {  
 **public static void** main(String[] args) {  
 */\*  
 自动装箱：直接把int类型的整数赋值给包装类。  
 Integer in = 1; 相当于 Integer in = new Integer(1)  
 \*/* Integer in = 1;

*/\*  
 自动拆箱：in是包装类，无法直接参与运算，可以自动转换为基本类型的数据，在参与计算。  
 in = in + 2;  
 in + 2; 就相当于 in.intValue() + 3 = 3;  
 in = in + 2；就相当于 in = new Integer(3) 自动装箱   
 \*/* in = in + 2;  
 System.***out***.println(in);  
  
 *// ArrayList集合无法直接存储整数，可以存储Integer包装类* ArrayList<Integer> list = **new** ArrayList<>();  
   
 list.add(1); *// 自动装箱：list.add(new Integer(1))* **int** a = list.get(9); *// 自动拆箱 list.get(0).intValue();* }  
}

5.4基本类型与字符串之间的转换

(1)基本数据类型转换为String

基本类型转换String总共有三种方式，这里只讲解最简单的一种方法：

基本类型直接与””相连接即可；如：34+””

1. String转换为对应的基本类型：

除了Character类之外，其他都有的包装类都具有parseXxx静态方法可以将字符串参数转化为对应的基本类型：

public static byte parseByte(String s) :将字符串参数转换为对应的byte基本类型。

public static short parseShort(String s) :将字符串参数转换为对应的short基本类型。

public static int parseInt(String s) :将字符串参数转换为对应的int基本类型。

public static long parseLong(String s) :将字符串参数转换为对应的long基本类型。

public static float parseFloat(String s) :将字符串参数转换为对应的float基本类型。

public static double parseDouble(String s) :将字符串参数转换为对应的double基本类型。

public static boolean parseBoolean(String s) :将字符串参数转换为对应的boolean基本类型。

/\*  
 基本类型与字符之间的转换：  
 基本类型 --> 字符串  
 1.基本类型数据的值+""最简单的方法（工作中常用）  
 2.使用包装类中的静态方法  
 static String toString(int i)：返回一个表示指定整数的 String 对象。  
 3.使用String类中的静态方法：  
 static String valueOf(int i):返回 int 参数的字符串表示i形式。

字符串 --> 基本类型  
 使用包装类的静态方法parseXxx("字符串")  
 Integer类： static int parseInt(String s)  
 Double类： Static double parseDouble(String s)  
 ...  
 \*/  
public class DemoString {  
 public static void main(String[] args) {  
  
 // 基本类型 --> 字符串  
 // 第一种 基本数据的值+"";  
 String s1 = 100+"";  
 System.out.println(s1+200); // 100200  
  
 // 第二种 使用静态方法 toString  
 String s2 = Integer.toString(100);  
 System.out.println(s2+200); // 100200  
  
 // 第三种 使用String中的 valueOf(int i)方法  
 String s3 = String.valueOf(100);  
 System.out.println(s3+200); //100200  
  
 // 字符串 --> 基本类型  
 int i = Integer.parseInt("100");  
 System.out.println(i+200); // 300 此时的i为基本类型int，代表数字，而非字符串  
  
 int i2 = Integer.parseInt("a");  
 System.out.println(i2); // NumberFormatException: For input string: "a",数字格式化异常。  
 }  
}