## 第1章 Java开发入门

一．填空题

1. 面向对象，SUN

2. JavaSE，JavaEE，JavaME

3.面向对象、跨平台性、支持多线程

4. JDK

5.bin目录

二．判断

1.对 2.错 3.错 4.对 5.错

三．选择

1.C 2.ABCD 3.D 4.A 5.D

四．简答

1．简答性、面向对象、安全性、跨平台性、支持多线程、分布性。

2. Java程序运行时，必须经过编译和运行两个步骤。首先将后缀名为.java的源文件进行编译，生成后缀名为.class的字节码文件，然后Java虚拟机将字节码文件进行解释执行，并将结果显示出来。

五．编程

HelloWorld.java

1 public class HelloWorld {

2 public static void main(String[] args) {

3 System.out.println("这是第一个Java程序!");

4 }

5 }

## 第2章 Java编程基础

一．填空题

1. class

2. 单行注释、多行注释、文档注释

3．基本数据、引用数据

4.&、&&、|、||

5.容器、0

二．判断

1. 错 2.错 3.对 4.对 5.对

三．选择

1. AC 2.A 3.B 4. D 5.C

四．简答

1. Java语言的八种基本数据类型有：byte字节型，占一个字节。short短整型，占两个字节。int整型，占4个字节。long长整型，占8个字节。float单精度浮点型，占4个字节。double双精度浮点型，占8个字节。char字符型，占两个字节。boolean型，表示逻辑值，有true和false两个值，分别占一个字节。

2. 在switch条件语句和循环语句中都可以使用break语句。当它出现在switch条件语句中时，作用是终止某个case并跳出switch结构。当它出现在循环语句中，作用是跳出循环语句，执行循环后面的代码；continue语句用在循环语句中，它的作用是终止本次循环，执行下一次循环。

五．编程

1. 参考答案

1. public class getSum {
2. public static void main(String[] args) {
3. int sum = 0;
4. for (int i = 1; i < 100; i++) {
5. if (i % 2 != 0)
6. sum += i;
7. }
8. System.out.println(sum);
9. }
10. }

2.参考答案

1. public class Example28 {
2. public static void main(String[] args) {
3. int[] arr = { 22，24，76，12，21，33 };
4. int max = getMax(arr);
5. System.out.println("max=" + max);
6. }
7. static int getMax(int[] arr) {
8. int max = arr[0];/
9. for (int x = 1; x < arr.length; x++) {
10. if (arr[x] > max) { /
11. max = arr[x];
12. }
13. }
14. return max;
15. }
16. }

## 第3章 面向对象（上）

一、填空题

1．封装、继承、多态

2．class

3．private、protected、public

4．static

5．private

二、判断题

1．错

2．错

3．对

4．对

5．对

三、选择题

1．D

2．B

3．D

4．A

5．B

四、简答题

1．封装、继承、多态

面向对象的特点主要可以概括为封装性、继承性和多态性。

其中封装是面向对象核心思想","将对象的属性和行为封装起来","不需要让外界知道具体实现细节","这就是封装思想。

继承性主要描述的是类与类之间的关系","通过继承","可以在无需重新编写原有类的情况下","对原有类的功能进行扩展。

多态性指的是在程序中允许出现重名现象","它指在一个类中定义的属性和方法被其它类继承后","它们可以具有不同的数据类型或表现出不同的行为","这使得同一个属性和方法在不同的类中具有不同的语义。

2．在Java中，定义在类中的变量被称为成员变量。定义在方法中的变量被称为局部变量。如果在某一个方法中定义的局部变量与成员变量同名，这种情况是允许的。此时，在方法中通过变量名访问到的是局部变量，而并非成员变量。

五、编程题

Student.java

1. public class Student {
2. private String sNo;
3. private String sName;
4. private String sSex;
5. private int sAge;
6. private int sJava;
7. //getter&setter方法
8. public String getsNo() {
9. return sNo;
10. }
11. public void setsNo(String sNo) {
12. this.sNo = sNo;
13. }
14. public String getsName() {
15. return sName;
16. }
17. public void setsName(String sName) {
18. this.sName = sName;
19. }
20. public String getsSex() {
21. return sSex;
22. }
23. public void setsSex(String sSex) {
24. this.sSex = sSex;
25. }
26. public int getsAge() {
27. return sAge;
28. }
29. public void setsAge(int sAge) {
30. this.sAge = sAge;
31. }
32. public int getsJava() {
33. return sJava;
34. }
35. public void setsJava(int sJava) {
36. this.sJava = sJava;
37. }
38. public Student(String sNo, String sName, String sSex, int sAge, int sJava) {
39. this.sNo = sNo;
40. this.sName = sName;
41. this.sSex = sSex;
42. this.sAge = sAge;
43. this.sJava = sJava;
44. }
45. }

Test.java

1. public class Test {
2. public static void main(String[] args) {
3. Student[] students = new Student[5];
4. students[0] = new Student("220110", "Tom", "男", 18, 85);
5. students[1] = new Student("220111", "Tohm", "男", 18, 82);
6. students[2] = new Student("220112", "Tomf", "女", 18, 79);
7. students[3] = new Student("220113", "WTom", "男", 18, 85);
8. students[4] = new Student("220114", "seTom", "男", 18, 90);
9. int sum = 0;
10. int average = 0;
11. int max = students[0].getsJava();
12. int min = students[0].getsJava();
13. for (Student student : students) {
14. int java = student.getsJava();
15. sum += java;
16. if (max < java) {
17. max = java;
18. }
19. if (min > java) {
20. min = java;
21. }
22. }
23. average = sum/(students.length);
24. System.out.println("Java语言的平均成绩是:"+average);
25. System.out.println("Java语言成绩的最大值是:"+max);
26. System.out.println("Java语言成绩的最小值是:"+min);
27. }

## 第4章 面向对象（下）

一、填空题

1．1

2．重写

3．final

4．instanceof

5．implements

二、判断题

1．对

2．对

3．错

4．对

5．错

三、选择题

1．B

2．D

3．B

4．D

5．C

四、简答题

1．概念：在Java中","类的继承是指在一个现有类的基础上去构建一个新的类","构建出来的新类被称作子类","现有类被称作父类","子类会自动拥有父类所有可继承的属性和方法。

好处：继承性主要描述的是类与类之间的关系","通过继承","可以无需重新编写原有类的情况下","对原有类的功能进行使用和扩展。

2．1、应用程序不必为每一个子类编写功能调用","只需要对抽象父类进行处理即可。大大提高程序的可复用性。

2、子类的功能可以被父类的方法或引用变量所调用","这叫向后兼容","可以提高可扩充性和可维护性。

3、使用多态可以解决项目中紧偶合的问题","提高程序的课扩展性.是OCP原则的一个具体的实现。

五、编程题

Employee.java

1. abstract class Employee{
2. private String name; //定义姓名name并私有化属性
3. private int month; //定义生日月份month并私有化属性
4. public Employee(){} //无参构造器
5. public Employee(String name,int month){ //有参构造方法
6. this.name = name; //给属性name初始化赋值
7. this.month = month; //给属性month初始化赋值
8. }
9. //获取属性name的方法
10. public String getName(){
11. return name; //返回name属性
12. }
13. //获取属性month的方法
14. public int getMonth(){
15. return month; //返回month属性
16. }
17. //给属性name赋初始值
18. public void setName(String name){
19. this.name = name; //本类中的属性name
20. }
21. //给属性month赋初始值
22. public void setMonth(int month){
23. this.month = month; //本类中的属性month
24. }
25. //创建getSalary()方法计算工资，参数month是月份，如果当月是员工生日，奖励100元
26. public double getSalary(int month){
27. double salary = 0; //定义工资变量
28. //判断当前月份是否是员 工的生日月份，如果是奖励100元
29. if(this.month == month){
30. salary = salary + 100;
31. return salary; //返回工资salary
32. }
33. }
34. }

SalariedEmployee.java

1. class SalariedEmployee extends Employee{
2. private double monthSalary; //封装monthSalary属性
3. public SalariedEmployee(){} //无参构造方法
4. //有参构造方法 参数 姓名 生日月份 月薪
5. public SalariedEmployee(String name,int month,double monthSalary){
6. super(name,month); //调用父类有参构造方法
7. this.monthSalary = monthSalary; //为属性monthSalary初始化赋值
8. }
9. //获取monthSalary的值
10. public double getMonthSalary(){
11. return monthSalary;
12. }
13. //给monthSalary赋值
14. public void setMonthSalary(double monthSalary){
15. this.monthSalary = monthSalary;
16. }
17. //覆盖父类中的方法
18. public double getSalary(int month){
19. //定义工资变量
20. double salary = monthSalary+super.getSalary(month);
21. return salary;
22. }
23. }

HourlyEmployee.java

1. class HourlyEmployee extends Employee{
2. private double hourlySalary; //定义属性hourlySalary每小时的工资
3. private int hours; //定义属性hours每月工作的小时数
4. public HourlyEmployee(){} //无参构造方法
5. //有参构造方法 参数 姓名 生日月份 每小时的工资 每月工作的小时数
6. public HourlyEmployee(String name,int month,double hourlySalary,int hours){
7. super(name,month); //调用父类有参构造方法
8. this.hourlySalary = hourlySalary ; //为属性hourlySalary初始化赋值
9. this.hours = hours; //为属性hours 初始化赋值
10. }
11. public double getHourlySalary(){ //获取hourlySalary的值
12. return hourlySalary;
13. }
14. public int getHours(){ //获取hours的值
15. return hours;
16. }
17. //定义set方法设置hourlySalary hours的值
18. public void setHourlySalary(double hourlySalary){
19. this.hourlySalary =hourlySalary;
20. }
21. public void setHourly(int hours){
22. this.hours = hours;
23. }
24. //覆盖父类方法
25. public double getSalary(int month){
26. if(hours < 0){ //如果工作小时数小于0 输出数据错误
27. System.out.println("数据错误");
28. return 0;
29. }
30. //小于160个小时的 按照每个月的工作小时数乘以每小时的工资
31. else if(hours <= 160)
32. return hourlySalary\*hours+super.getSalary(month);
33. //超出160个小时的小时数 按照1.5倍计算
34. else return hourlySalary\*160+hourlySalary\*1.5\*(hours- 160)+super.getSalary(month);
35. }
36. }

SalesEmployee.java

1. class SalesEmployee extends Employee{
2. private double sales ; //定义销售额sales
3. private double rate; //定义提成率rate
4. public SalesEmployee(){}
5. public SalesEmployee(String name,int month,double sales,double rate){
6. super(name,month);
7. this.sales = sales;
8. this.rate = rate;
9. }
10. public double getSales(){
11. return sales;
12. }
13. public double getRate(){
14. return rate;
15. }
16. public void setSales(double sales){
17. this.sales = sales;
18. }
19. public void setRate(double rate){
20. this.rate = rate;
21. }
22. public double getSalary(int month){
23. return this.getSales()\*(1+this.getRate())+super.getSalary(month);
24. }
25. }

BasePlusSalesEmployee.java

1. class BasePlusSalesEmployee extends SalesEmployee{
2. private double baseSalary; //定义基础工资baseSalary
3. //无参构造方法
4. public BasePlusSalesEmployee(){}
5. //有参构造方法
6. public BasePlusSalesEmployee(String name,int month,double sales,double rate,double baseSalary){
7. super(name,month,sales,rate);
8. this.baseSalary = baseSalary;
9. }
10. //get/set方法对私有属性的调用和设置
11. public double gatBaseSalary(){
12. return baseSalary;
13. }
14. public void setBaseSalary(){
15. this.baseSalary = baseSalary;
16. }
17. public double getSalary(int month){
18. return baseSalary+super.getSalary(month);
19. }
20. }

Test.java

1. //定义一个测试类
2. public class Test{
3. public static void main(String[] args){
4. //声明一个Employee类型的数组，并创建不同子类型的对象
5. Employee[] employee = {new SalariedEmployee(“张三”,1,6000),new HourlyEmployee(“李 四”,2,50,180),new SalesEmployee(“王 五”,3,6500,0.15),new BasePlusSalesEmployee(“赵 六”,4,5000,0.15,2000)};
6. //打印每个员工的工资
7. for(int i = 0; i < employee.length ;i++)
8. System.out.println(Math.round(employee[i].getSalary(10)));
9. }
10. }

## 第5章 Java API

一．填空

1.String StringBuffer

2. length()

3. Math.abs();

4. 随机产生数字

5. Long字段 标准Java计算时代（就是1970年1月1日开始） 纳秒数

二．判断

1.错 2.错 3.错 4.对 5.对

三．选择

1.B 2.C 3.A 4.A 5.A

四．简答

1. StringBuffer类和StringBuilder类的对象都可以被多次修改，并不产生新的未使用对象，StringBuilder类是JDK5中新加的类，它与StringBuffer之间最大不同在于StringBuilder的方法是线程安全的，也就是说StringBuffer不能被同步访问，而StringBuilder可以。

2．

|  |  |
| --- | --- |
| **基本数据类型** | **对应的包装类** |
| byte | Byte |
| char | Character |
| int | Integer |
| short | Short |
| long | Long |
| float | Float |
| double | Double |
| boolean | Boolean |

五．编程

1. 参考答案

1. public static void main(String args) {
2. for(int i=0;i<10;i++){
3. System.out.println(new Random().nextInt(100));
4. }
5. }

2.参考答案

1. public class Test01 {
2. public static void main(String[] args) {
3. String str = "HelloWorld";
4. // 字符串转成char数组
5. char[] ch = str.toCharArray();
6. StringBuffer buffer = new StringBuffer();
7. for (int i = str.length() - 1; i >= 0; i--) {
8. if (ch[i] >= 'A' && ch[i] <= 'Z') {
9. buffer.append(String.valueOf(ch[i]).toLowerCase());
10. } else if (ch[i] >= 'a' && ch[i] <= 'z') {
11. buffer.append(String.valueOf(ch[i]).toUpperCase());
12. }
13. }
14. System.out.println(buffer.toString());
15. }
16. }

## 第6章 集合类

一．填空

1. Collection

2.hashNext() next()

3. ArrayList、LinkedList，HashSet、TreeSet，HashMap、TreeMap

4. Key、Value

5. 数组

二．判断

1.错 2.对 3.对 4.对 5.错

三．选择

1.BC

2.B

3.ABCD

4.D

5.C

四．简答

1. List的特点是元素有序、可重复。List接口的主要实现类有ArrayList和LinkedList。Set的特点是元素无序、不可重复。Set接口的主要实现类有HashSet和TreeSet。Map的特点是存储的元素是键(Key)、值(Value)映射关系，元素都是成对出现的。Map接口的主要实现类有HashMap和TreeMap。

2. 由于ArrayList集合的底层是使用一个数组来保存元素，在增加或删除指定位置的元素时，会导致创建新的数组，效率比较低，因此不适合做大量的增删操作。但这种数组的结构允许程序通过索引的方式来访问元素，因此使用ArrayList集合查找元素很便捷。

五．编程

1.参考答案

1. import java.util.\*;
2. class Student {
3. private int age;
4. private String name;
5. public Student(int age, String name) {
6. this.age = age;
7. this.name = name;
8. }
9. public String toString() {
10. return age + ":" + name;
11. }
12. public int hashCode() {
13. return name.hashCode() + age;
14. }
15. public boolean equals(Object obj) {
16. if (this == obj)
17. return true;
18. if (!(obj instanceof Student))
19. return false;
20. Student stu = (Student) obj;
21. return this.name.equals(stu.name) && this.age == stu.age;
22. }
23. }
24. public class Test {
25. public static void main(String[] args) {
26. HashSet<Student> hs = new HashSet<Student>();
27. hs.add(new Student(18, "zhangsan"));
28. hs.add(new Student(20, "lisa"));
29. hs.add(new Student(20, "lisa"));
30. System.out.println(hs);
31. }
32. }

2.参考答案

1. interface Generic<T>{
2. public abstract void get(T t){}
3. }
4. class Generic<T> implements Generic{
5. public void get(T t){}
6. }

## 第7章 IO（输入输出）

一．填空

1.字节流、字符流

2. java.io

3．二进制(字节)

4．BufferedInputStream BufferedOutputStream

5. InputStreamReader OutputStreamWriter

二．判断

1.错 2.对 3.错 4.错

三．选择

1．AB 2.C 3.A 4.C 5.C

四．简答

1. 字节流的两个基类是InputStream和OutputStream，字符流的两个基类是Reader和Writer，它们都是Object类的直接子类，字节流是处理以8位字节为基本单位的字节流类；Reader和Writer类是专门处理16位字节的字符流类。

2. InputStreamReader是Reader的子类，它可以将一个字节输入流转换成字符输入流，方便直接读取字符。OutputStreamWriter是Writer的子类，它可以将一个字节输出流转换成字符输出流，方便直接写入字符。

五．编程

1.参考答案

1. import java.io.\*;
2. public class Test01 {
3. public static void main(String[] args) throws Exception {
4. // 字节流拷贝
5. FileInputStream in = new FileInputStream("E:/src.txt");
6. FileOutputStream out = new FileOutputStream("E:/des1.txt");
7. byte[] buf = new byte[1024];
8. int len;
9. while ((len = in.read(buf)) != -1) {
10. out.write(buf, 0, len);
11. }
12. in.close();
13. out.close();
14. // 字符流拷贝
15. BufferedReader bf = new BufferedReader(new FileReader("E:/src.txt"));
16. BufferedWriter bw = new BufferedWriter(new FileWriter("E:/des2.txt"));
17. String str;
18. while ((str = bf.readLine()) != null) {
19. bw.write(str);
20. bw.newLine();
21. }
22. bf.close();
23. bw.close();
24. }
25. }

## 第8章 多线程

一．填空

1. Thread Runnable

2. 新建状态（New）、就绪状态（Runnable）、运行状态（Running）

3. start()、run()方法

4. sleep()

5. synchronized

二．判断

1.错 2.错 3.对 4.对 5.对

三．选择

1.C 2.C 3.C 4.D 5.BC

四．简答

1．一种是继承java.lang包下的Thread类，覆写Thread类的run()方法，在run()方法中实现运行在线程上的代码。

new Thread() {

public void run(){}

}.start();

另一种就是实现java.lang.Runnable接口，同样是在run()方法中实现运行在线程上的代码。

class MyThread implements Runnable{

public void run(){}

}

2. 同步代码块的作用是控制线程，保证同步代码块中只能有一个线程在运行，保证了多线程操作数据的安全性。

五．编程

1.参考答案

1. public class Test01 {
2. public static void main(String[] args) {
3. Teacher t = new Teacher();
4. new Thread(t, "陈老师").start();
5. new Thread(t, "高老师").start();
6. new Thread(t, "李老师").start();
7. }
8. }
9. class Teacher implements Runnable {
10. private int notes = 50;
11. public void run() {
12. while (true) {
13. dispatchNotes();
14. if (notes <= 0) {
15. break;
16. }
17. }
18. }
19. private synchronized void dispatchNotes() {
20. if (notes > 0) {
21. try {
22. Thread.sleep(10); // 经过的线程休眠10毫秒
23. } catch (InterruptedException e) {
24. e.printStackTrace();
25. }
26. System.out.println(Thread.currentThread().getName() + "---发出的苹果"
27. + notes--);
28. }
29. }
30. }

## 第9章 网络编程

一、填空题

1．UDP

2．TCP

3．DatagramSocket

4．ServerSocket

5．IO流

二、判断题

1．对

2．对

3．错

4．对

5．错

三、选择题

1．D

2．A

3．D

4．B

5．C

四、简答题

1．UDP通信与TCP通信的区别在于，UDP中只有发送端和接收端，不区分客户端与服务器端，计算机之间可以任意地发送数据；而TCP通信是严格区分客户端与服务器端的，在通信时，必须先由客户端去连接服务器端才能实现通信，服务器端不可以主动连接客户端，并且服务器端程序需要事先启动，等待客户端的连接。

2．在TCP连接中必须要明确客户端与服务器端，由客户端向服务器端发出连接请求，每次连接的创建都需要经过“三次握手”。第一次握手，客户端向服务器端发出连接请求，等待服务器确认；第二次握手，服务器端向客户端回送一个响应，通知客户端收到了连接请求；第三次握手，客户端再次向服务器端发送确认信息，确认连接。

五、编程题

AskServer.java

1. /\*\*
2. \*
3. \* 在线客服咨询人员
4. \*
5. \*/
6. public class AskServer {
7. public static void main(String[] args) {
8. //创建DatagramSocket，发送接收数据都依赖他
9. DatagramSocket socket = null;
10. try {
11. socket = new DatagramSocket(8888);
12. Scanner input = new Scanner(System.in);
13. while (true) {
14. //准备一个空的数据包，用来接收数据
15. byte[] buf = new byte[1024];
16. DatagramPacket packet = new DatagramPacket(buf, buf.length);
17. //接收数据使用空的数据包
18. socket.receive(packet);
19. //输出信息
20. String info = new String(packet.getData(), 0, packet.getLength());
21. System.out.println("客户端请求：" + info);
22. //判断是否退出
23. if ("bye".equals(info)) {
24. break;
25. }
26. //发送数据
27. String result = input.nextLine();
28. byte[] buf2 = result.getBytes();
29. DatagramPacket packet2 = new DatagramPacket(buf2, buf2.length,packet.getAddress(), packet.getPort());
30. socket.send(packet2);
31. }
32. } catch (SocketException e) {
33. e.printStackTrace();
34. } catch (IOException e) {
35. e.printStackTrace();
36. } finally {
37. //关闭socket
38. socket.close();
39. }
40. }
41. }

AskClient.java

1. /\*\*
2. \*
3. \* 在线客服客户
4. \*
5. \*/
6. public class AskClient {
7. public static void main(String[] args) {
8. //创建DatagramSocket，发送接收数据都依赖他
9. DatagramSocket socket = null;
10. try {
11. socket = new DatagramSocket(9999);
12. Scanner input = new Scanner(System.in);
13. while (true){
14. //准备要发送的数据
15. String str = input.nextLine();//bye
16. //使用数据包把数据封装起来
17. byte[] buf = str.getBytes();
18. DatagramPacket packet = new DatagramPacket(buf, buf.length,
19. InetAddress.getByName("192.168.1.252"), 8888);
20. //发送数据包
21. socket.send(packet);
22. //判断是否退出
23. if ("bye".equals(str)) {
24. break;
25. }
26. //接收数据
27. byte[] buf2 = new byte[1024];
28. DatagramPacket packet2 = new DatagramPacket(buf2, buf2.length);
29. socket.receive(packet2);
30. System.out.println("服务器端反馈："+
31. new String(packet2.getData(), 0, packet2.getLength()));
32. }
33. } catch (SocketException e) {
34. e.printStackTrace();
35. } catch (IOException e) {
36. e.printStackTrace();
37. } finally {
38. //关闭socket
39. socket.close();
40. }
41. }
42. }

## 第10章 JDBC

一、填空题

1．java.sql.DriverManager

2．classpath

3．update

4．预编译

5．1

二、判断题

1．错

2．错

3．对

4．错

5．错

三、选择题

1．A

2．A

3．B

4．B

5．C

四、简答题

1.参考答案：

（1）加载并注册数据库驱动；

（2）通过DriverManager获取数据库连接；

（3）通过Connection对象获取Statement对象；

（4）使用Statement执行SQL语句；

（5）操作ResultSet结果集；

（6）回收数据库资源。

2． 参考答案

所谓预编译,就是说当相同的SQL语句再次执行时,数据库只需使用缓冲区中的数据,而不需要对SQL语句再次编译,从而有效提高数据的访问效率。

## 第11章 GUI

一、填空题

1．CENTER

2．容器边界

3．setLayout()

4．GridBagConstraints

二、判断题

1．对

2．错

3．错

4．对

5．对

三、选择题

1．D

2．A

3．A

4．C

5．C

四、简答题

1．参考答案

  1、创建GridbagLayout布局管理器，并使容器采用该布局管理器。

  2、创建GridBagContraints对象(布局约束条件)，并设置该对象的相关属性。

  3、调用GridBagLayout对象的setConstraints()方法建立GridBagConstraints对象和受控组件之间的关联。

  4、向容器中添加组件。

2． 参考答案

（1）创建一个JMenuBar菜单栏对象，将其放置在JFrame窗口的顶部。

（2）创建JMenu菜单对象，将其添加到JMenuBar菜单栏中。

（3）创建JMenuItem菜单项，将其添加到JMenu菜单中。

五、编程题

CalTest.java

1. import javax.swing.\*;
2. import java.awt.\*;
3. import java.awt.event.ActionEvent;
4. import java.awt.event.ActionListener;
5. public class CalTest extends JFrame{
6. private TextField firText,secText,reText;
7. private JLabel eqLabel;
8. private JComboBox jbox;
9. private String[] str;
10. private Button calb,clearb,exitb;
11. private Panel p,p2;
12. public static void main(String[] args) {
13. new CalTest("运算器");
14. }
15. public CalTest(){}
16. public CalTest(String title){
17. setTitle(title);
18. setLocation(500, 200);
19. setDefaultCloseOperation(3);
20. setResizable(false);
21. setLayout(new FlowLayout());
22. init();
23. pack();
24. setVisible(true);
25. }
26. public void init(){
27. p = new Panel();
28. p.setSize(300, 300);
29. firText = new TextField(5);
30. secText = new TextField(5);
31. reText = new TextField(5);
32. reText.setEditable(false);
33. eqLabel = new JLabel("=");
34. str = new String[]{"+","-","\*","/"};
35. jbox = new JComboBox(str);
36. p.add(firText);
37. p.add(jbox);
38. p.add(secText);
39. p.add(eqLabel);
40. p.add(reText);
41. add(p);
42. p2 = new Panel();
43. calb = new Button("cal");
44. calb.addActionListener(new ActionListener() {
45. public void actionPerformed(ActionEvent arg0) {
46. //getSelectedItem()返回JComboBox所选项，返回Objection类型
47. String sign = (String) jbox.getSelectedItem();
48. if(sign.equals("+")){
49. int num = Integer.parseInt(firText.getText());
50. int num2 = Integer.parseInt(secText.getText());
51. int add = num+num2;
52. reText.setText(add+"");
53. }else if(sign.equals("-")){
54. int num = Integer.parseInt(firText.getText());
55. int num2 = Integer.parseInt(secText.getText());
56. int sup = num-num2;
57. reText.setText(sup+"");
58. }else if(sign.equals("\*")){
59. int num = Integer.parseInt(firText.getText());
60. int num2 = Integer.parseInt(secText.getText());
61. int sel = num\*num2;
62. reText.setText(sel+"");
63. }else{
64. int num = Integer.parseInt(firText.getText());
65. int num2 = Integer.parseInt(secText.getText());
66. int sele = num/num2;
67. reText.setText(sele+"");
68. }
69. }
70. });
71. clearb = new Button("clear");
72. clearb.addActionListener(new ActionListener() {
73. public void actionPerformed(ActionEvent arg0) {
74. firText.setText(null);
75. secText.setText(null);
76. reText.setText(null);
77. }
78. });
79. exitb = new Button("exit");
80. exitb.addActionListener(new ActionListener() {
81. public void actionPerformed(ActionEvent arg0) {
82. System.exit(0);
83. }
84. });
85. p2.add(calb);
86. p2.add(clearb);
87. p2.add(exitb);
88. add(p2, BorderLayout.SOUTH);
89. }
90. }

## 第12章 反射

一、填空题

1．动态

2．newInstance()

3．Constructor，Field，Method

4．getInterfaces()

5．java.lang.Class

二、判断题

1．对

2．对

3．对

4．错

5．对

三、选择题

1．D

2．C

3．B

4．AC

5．C

四、简答题

1．Java的反射（reflection）机制是指在程序的运行状态中，可以构造任意一个类的对象，可以得到任意一个对象所属的类的信息，可以调用任意一个类的成员变量和方法，可以获取任意一个对象的属性和方法。这种动态获取程序信息以及动态调用对象的功能称为Java语言的反射机制。

2．实例化Class对象共有以下三种方式：

（1）根据类名获取：类名.class；

（2）根据对象获取：对象.getClass()；

（3）根据全限定类名获取：Class.forName(“全限定类名”)。