· 1.

· (单选题)下列代码编译和运行的结果是：（d）。

|  |  |
| --- | --- |
| · 01 | public class Foo { |
| · 02 | public static void test(String str) { |
| · 03 | int check = 3; |
| · 04 | if (check == str.length()) { |
| · 05 | System.out.print(str.charAt(check -= 1) + ", "); |
| · 06 | } else { |
| · 07 | System.out.print(str.charAt(0) + ", "); |
| · 08 | } |
| · 09 | } |
| · 10 | public static void main(String[] args) { |
| · 11 | test("four"); |
| · 12 | test("tea"); |
| · 13 | test("to"); |
| · 14 | } |
| · 15 | } |

· A.

f, a, t,

B.

f, t, t,

C.

编译失败

D.

抛出运行时异常

· 2.

· (单选题)下列代码输出的结果是:（d）。

|  |  |
| --- | --- |
| · 1 | String name="NASDAQ欢迎你"; |
| · 2 | String newName=name.substring(4,7); |
| · 3 | System.out.println(newName); |

· A.

DAQ

B.

SDAQ欢

C.

AQ欢迎

D.

AQ欢

· 3.

· (单选题)下列代码的运行结果是：（a）。

|  |  |
| --- | --- |
| · 1 | String test = "tarena7cstp8info9"; |
| · 2 | String[] tokens = test.split("\\d"); |
| · 3 | for (String s : tokens) { |
| · 4 | System.out.print(s + " "); |
| · 5 | } |

· A.

tarena cstp info

B.

7 8 9

C.

tarena7cstp8info9

D.

tarena7 cstp8 info9

· 4.

· (单选题)一个类中方法如下：

|  |  |
| --- | --- |
| · 1 | public String getInfo() { |
| · 2 | StringBuffer buffer = new StringBuffer(); |
| · 3 | buffer.append('<'); |
| · 4 | buffer.append(this.name); |
| · 5 | buffer.append('>'); |
| · 6 | return buffer.toString(); |
| · 7 | } |

· 下列说法正确的是：（b）。

· A.

这段代码线程不安全

B.

把程序中的StringBuffer替换成StringBuilder，返回结果无变化

C.

把程序中的StringBuffer替换成StringBuilder，不能提高程序性能

D.

为了提高程序性能，getInfo方法的实现代码应改为：return "<"+ this.name + ">";

· 5.

· (单选题)已知类Emp的定义如下：

|  |  |
| --- | --- |
| · 1 | public class Emp { |
| · 2 | private String name; |
| · 3 | public Emp(String name) { |
| · 4 | this.name = name; |
| · 5 | } |
| · 6 | } |

· 如果两个引用的Emp对象的属性name对应的字符序列相同，那么使用equals方法比较这两个引用则返回true。

请问下列选项中在Emp类中覆盖equals方法正确的是：（b）。

· A.

|  |  |
| --- | --- |
| 01 | public boolean equals(Object obj) { |

|  |  |
| --- | --- |
| 02 | if (this == obj) |

|  |  |
| --- | --- |
| 03 | return true; |

|  |  |
| --- | --- |
| 04 | if (obj == null) |

|  |  |
| --- | --- |
| 05 | return false; |

|  |  |
| --- | --- |
| 06 | if (getClass() != obj.getClass()) |

|  |  |
| --- | --- |
| 07 | return false; |

|  |  |
| --- | --- |
| 08 | Emp other = (Emp) obj; |

|  |  |
| --- | --- |
| 09 | if (name == null) { |

|  |  |
| --- | --- |
| 10 | if (other.name != null) |

|  |  |
| --- | --- |
| 11 | return false; |

|  |  |
| --- | --- |
| 12 | } else if (name!=other.name) |

|  |  |
| --- | --- |
| 13 | return false; |

|  |  |
| --- | --- |
| 14 | return true; |

|  |  |
| --- | --- |
| 15 | } |

B.

|  |  |
| --- | --- |
| 01 | public boolean equals(Object obj) { |

|  |  |
| --- | --- |
| 02 | if (this == obj) |

|  |  |
| --- | --- |
| 03 | return true; |

|  |  |
| --- | --- |
| 04 | if (obj == null) |

|  |  |
| --- | --- |
| 05 | return false; |

|  |  |
| --- | --- |
| 06 | if (getClass() != obj.getClass()) |

|  |  |
| --- | --- |
| 07 | return false; |

|  |  |
| --- | --- |
| 08 | Emp other = (Emp) obj; |

|  |  |
| --- | --- |
| 09 | if (name == null) { |

|  |  |
| --- | --- |
| 10 | if (other.name != null) |

|  |  |
| --- | --- |
| 11 | return false; |

|  |  |
| --- | --- |
| 12 | } else if (!name.equals(other.name)) |

|  |  |
| --- | --- |
| 13 | return false; |

|  |  |
| --- | --- |
| 14 | return true; |

|  |  |
| --- | --- |
| 15 | } |

C.

|  |  |
| --- | --- |
| 01 | public boolean equals(Object obj) { |

|  |  |
| --- | --- |
| 02 | if (this == obj) |

|  |  |
| --- | --- |
| 03 | return true; |

|  |  |
| --- | --- |
| 04 | if (obj == null) |

|  |  |
| --- | --- |
| 05 | return false; |

|  |  |
| --- | --- |
| 06 | if (getClass() != obj.getClass()) |

|  |  |
| --- | --- |
| 07 | return false; |

|  |  |
| --- | --- |
| 08 | if (name == null) { |

|  |  |
| --- | --- |
| 09 | if (obj.name != null) |

|  |  |
| --- | --- |
| 10 | return false; |

|  |  |
| --- | --- |
| 11 | } else if (name==obj.name) |

|  |  |
| --- | --- |
| 12 | return false; |

|  |  |
| --- | --- |
| 13 | return true; |

|  |  |
| --- | --- |
| 14 | } |

D.

|  |  |
| --- | --- |
| 01 | public boolean equals(Object obj) { |

|  |  |
| --- | --- |
| 02 | if (this == obj) |

|  |  |
| --- | --- |
| 03 | return true; |

|  |  |
| --- | --- |
| 04 | if (obj == null) |

|  |  |
| --- | --- |
| 05 | return false; |

|  |  |
| --- | --- |
| 06 | if (getClass() != obj.getClass()) |

|  |  |
| --- | --- |
| 07 | return false; |

|  |  |
| --- | --- |
| 08 | if (name == null) { |

|  |  |
| --- | --- |
| 09 | if (obj.name != null) |

|  |  |
| --- | --- |
| 10 | return false; |

|  |  |
| --- | --- |
| 11 | } else if (!name.equals(obj.name)) |

|  |  |
| --- | --- |
| 12 | return false; |

|  |  |
| --- | --- |
| 13 | return true; |

|  |  |
| --- | --- |
| 14 | } |

· 6.

· (单选题)下列代码的运行结果是：（a）。

|  |  |
| --- | --- |
| · 1 | public class JDK5 { |
| · 2 | public static void main(String[] args) { |
| · 3 | Double i = 12.3; |
| · 4 | double j = 12.3; |
| · 5 | System.out.println("It is " + (j == i) + " that j==i"); |
| · 6 | } |
| · 7 | } |

· A.

It is true that j==i

B.

It is false that j==i

C.

运行时抛出异常

D.

编译出错

· 7.

· (单选题)使用Date获取明天的日期-时间，下列选项正确的是：（c）。

· A.

|  |  |
| --- | --- |
| 1 | Date date = new Date(new Now()); |

|  |  |
| --- | --- |
| 2 | long time = date.getTime(); |

|  |  |
| --- | --- |
| 3 | date.setTime(time + 24 \* 60 \* 60 \* 1000) |

B.

|  |  |
| --- | --- |
| 1 | Date date = new Date(new Now()); |

|  |  |
| --- | --- |
| 2 | long time = date.getTime(); |

|  |  |
| --- | --- |
| 3 | date.setTime(time + 24 \* 60 \* 60); |

C.

|  |  |
| --- | --- |
| 1 | Date date = new Date(); |

|  |  |
| --- | --- |
| 2 | long time = date.getTime(); |

|  |  |
| --- | --- |
| 3 | date.setTime(time + 24 \* 60 \* 60 \* 1000); |

D.

|  |  |
| --- | --- |
| 1 | Date date = new Date(); |

|  |  |
| --- | --- |
| 2 | long time = date.getTime(); |

|  |  |
| --- | --- |
| 3 | date.setTime(time + 24 \* 60 \* 60); |

· 8.

· (单选题)把当前日期-时间转换为字符串“2014-04-15”的形式，下列选项正确的是：（c）。

· A.

|  |  |
| --- | --- |
| 1 | SimpleDateFormat sdf = new SimpleDateFormat("yyyy-mm-dd"); |

|  |  |
| --- | --- |
| 2 | Date date = new Date(); |

|  |  |
| --- | --- |
| 3 | String dateStr = sdf.format(date); |

B.

|  |  |
| --- | --- |
| 1 | SimpleDateFormat sdf = new SimpleDateFormat("yyyy-mm-dd"); |

|  |  |
| --- | --- |
| 2 | Date date = new Date(); |

|  |  |
| --- | --- |
| 3 | String dateStr = sdf.parse(date); |

C.

|  |  |
| --- | --- |
| 1 | SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd"); |

|  |  |
| --- | --- |
| 2 | Date date = new Date(); |

|  |  |
| --- | --- |
| 3 | String dateStr = sdf.format(date); |

D.

|  |  |
| --- | --- |
| 1 | SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd"); |

|  |  |
| --- | --- |
| 2 | Date date = new Date(); |

|  |  |
| --- | --- |
| 3 | String dateStr = sdf.parse(date); |

9.

· (单选题)计算2013年的每一个月各有多少天，并以“某年某月有多少天”的形式输出（例如：2013年1月有31天），

下列选项正确的是：（）。

· A.

|  |  |
| --- | --- |
| 01 | public static void main(String[] args) { |

|  |  |
| --- | --- |
| 02 | Calendar c = Calendar.getInstance(); |

|  |  |
| --- | --- |
| 03 | SimpleDateFormat sdf = new SimpleDateFormat("yyyy年M月"); |

|  |  |
| --- | --- |
| 04 | c.set(Calendar.YEAR, 2013); |

|  |  |
| --- | --- |
| 05 | c.set(Calendar.MONTH, Calendar.JANUARY); |

|  |  |
| --- | --- |
| 06 | c.set(Calendar.DATE, 1); |

|  |  |
| --- | --- |
| 07 | for (int i = 0; i < 12; i++) { |

|  |  |
| --- | --- |
| 08 | System.out.println(sdf.format(c.getTime()) + "有" + c.getActualMaximum(Calendar.MONTH) + "天"); |

|  |  |
| --- | --- |
| 09 | c.add(Calendar.MONTH, 1); |

|  |  |
| --- | --- |
| 10 | } |

|  |  |
| --- | --- |
| 11 | } |

B.

|  |  |
| --- | --- |
| 01 | public static void main(String[] args) { |

|  |  |
| --- | --- |
| 02 | Calendar c = Calendar.getInstance(); |

|  |  |
| --- | --- |
| 03 | SimpleDateFormat sdf = new SimpleDateFormat("yyyy年M月"); |

|  |  |
| --- | --- |
| 04 | c.set(Calendar.YEAR, 2013); |

|  |  |
| --- | --- |
| 05 | c.set(Calendar.MONTH, Calendar.JANUARY); |

|  |  |
| --- | --- |
| 06 | c.set(Calendar.DATE, 1); |

|  |  |
| --- | --- |
| 07 | for (int i = 0; i < 12; i++) { |

|  |  |
| --- | --- |
| 08 | System.out.println(sdf.format(c.getTime()) + "有" + c.getActualMaximum(Calendar.DATE) + "天"); |

|  |  |
| --- | --- |
| 09 | c.add(Calendar.MONTH, 1); |

|  |  |
| --- | --- |
| 10 | } |

|  |  |
| --- | --- |
| 11 | } |

C.

|  |  |
| --- | --- |
| 01 | public static void main(String[] args) { |

|  |  |
| --- | --- |
| 02 | Calendar c = Calendar.getInstance(); |

|  |  |
| --- | --- |
| 03 | SimpleDateFormat sdf = new SimpleDateFormat("yyyy年M月"); |

|  |  |
| --- | --- |
| 04 | c.set(Calendar.YEAR, 2013); |

|  |  |
| --- | --- |
| 05 | c.set(Calendar.MONTH, Calendar.JANUARY); |

|  |  |
| --- | --- |
| 06 | c.set(Calendar.DATE, 1); |

|  |  |
| --- | --- |
| 07 | for (int i = 0; i < 12; i++) { |

|  |  |
| --- | --- |
| 08 | System.out.println(sdf.format(c.getTime()) + "有" + c.getActualMaximum(Calendar.DATE) + "天"); |

|  |  |
| --- | --- |
| 09 | c.addMonth(Calendar.MONTH, 1); |

|  |  |
| --- | --- |
| 10 | } |

|  |  |
| --- | --- |
| 11 | } |

D.

|  |  |
| --- | --- |
| 01 | public static void main(String[] args) { |

|  |  |
| --- | --- |
| 02 | Calendar c = Calendar.getInstance(); |

|  |  |
| --- | --- |
| 03 | SimpleDateFormat sdf = new SimpleDateFormat("yyyy年M月"); |

|  |  |
| --- | --- |
| 04 | c.set(Calendar.YEAR, 2013); |

|  |  |
| --- | --- |
| 05 | c.set(Calendar.MONTH, Calendar.JANUARY); |

|  |  |
| --- | --- |
| 06 | c.set(Calendar.DATE, 1); |

|  |  |
| --- | --- |
| 07 | for (int i = 0; i < 12; i++) { |

|  |  |
| --- | --- |
| 08 | System.out.println(sdf.format(c.getTime()) + "有" + c.getActualMaximum(Calendar.MONTH) + "天"); |

|  |  |
| --- | --- |
| 09 | c.addMonth(Calendar.MONTH, 1); |

|  |  |
| --- | --- |
| 10 | } |

|  |  |
| --- | --- |
| 11 | } |

· 10.

· (单选题)请看下列代码：

|  |  |
| --- | --- |
| · 01 | public class CreateFile { |
| · 02 | public static void main(String[] args) { |
| · 03 | try { |
| · 04 | File directory = new File("d"); |
| · 05 | File file = new File(directory, "f"); |
| · 06 | if (!file.exists()) { |
| · 07 | file.createNewFile(); |
| · 08 | } |
| · 09 | } catch (IOException e) { |
| · 10 | e.printStackTrace(); |
| · 11 | } |
| · 12 | } |
| · 13 | } |

· 如果当前目录下不存在d目录，下列说法正确的是：（）。

· A.

代码file.createNewFile();不会被执行到

B.

抛出运行时异常

C.

代码File directory = new File("d");行，将在文件系统创建目录 "d"

D.

代码file.createNewFile();行，将在文件系统中创建目录 "d" 和文件 "f"

正确答案：B

解析：本题考查目录和文件的创建。本题代码想要实现在目录d下创建文件f。由于目录d不存在会抛出异常java.io.IOException: 系统找不到指定的路径。

· 11.

· (单选题)在Java中，要判断D盘下是否存在文件hello.txt,应该使用的判断语句是：（）。

· A.

if(new File("d:hello.txt") .exists()==1){}

B.

if(File.exists("d:hello.txt")==1) {}

C.

if(new File("d:/hello.txt") .exists( ) ) {}

D.

if(File.exists("d:/hello.txt")){}

正确答案：C

解析：本题考查如何判断文件是否存在。File类的exists()判断此抽象路径名表示的文件或目录是否存在，该方法的返回值是boolean类型，存在则返回true，否则返回false

· 12.

· (单选题)请看下列代码：

Map<String, Integer> map = new HashMap<String, Integer>();

map.put("G1", 50);

map.put("G2", 100);

map.put("G3", 300);

<插入代码>

for (Map.Entry<String, Integer> per : set) {

System.out.println(per.getKey() + ":" + per.getValue());

}

· A.

Set<Map.Entry<String, Integer>> set = map.entrySet();

B.

Set<Map.Entry<String, Integer>> set = map.keySet();

C.

Set<String> set = map.entrySet();

D.

Set<String> set = map.keySet();

· 13.

· (单选题)下面的代码用于输出字符数组ch中每个字符出现的次数，应该填入的代码是:

|  |  |
| --- | --- |
| · 1 | public static void main(String[] args) { |
| · 2 | char[] ch = { 'a', 'c', 'a', 'b', 'c', 'b' }; |
| · 3 | HashMap map = new HashMap(); |
| · 4 | for (int i = 0; i < ch.length; i++) { |
| · 5 | <        填入代码           > |
| · 6 | }         System.out.println(map); |
| · 7 | } |

· A.

|  |  |
| --- | --- |
| 1 | if (map.contains(ch[i])) { |

|  |  |
| --- | --- |
| 2 | map.add(ch[i], (Integer) map.get(ch[i]) + 1); |

|  |  |
| --- | --- |
| 3 | } else { |

|  |  |
| --- | --- |
| 4 | map.add(ch[i], 1); |

|  |  |
| --- | --- |
| 5 | } |

B.

|  |  |
| --- | --- |
| 1 | if (map.contains(ch[i])) { |

|  |  |
| --- | --- |
| 2 | map.put(ch[i], (Integer) map.get(ch[i]) + 1); |

|  |  |
| --- | --- |
| 3 | } else { |

|  |  |
| --- | --- |
| 4 | map.put(ch[i], 1); |

|  |  |
| --- | --- |
| 5 | } |

C.

|  |  |
| --- | --- |
| 1 | if (map.containsKey(ch[i])) { |

|  |  |
| --- | --- |
| 2 | map.add(ch[i], (Integer) map.get(ch[i]) + 1); |

|  |  |
| --- | --- |
| 3 | } else { |

|  |  |
| --- | --- |
| 4 | map.add(ch[i], 1); |

|  |  |
| --- | --- |
| 5 | } |

D.

|  |  |
| --- | --- |
| 1 | if (map.containsKey(ch[i])) { |

|  |  |
| --- | --- |
| 2 | map.put(ch[i], (Integer) map.get(ch[i]) + 1); |

|  |  |
| --- | --- |
| 3 | } else { |

|  |  |
| --- | --- |
| 4 | map.put(ch[i], 1); |

|  |  |
| --- | --- |
| 5 | } |

· 14.

· (单选题)请看下列代码：

|  |  |
| --- | --- |
| · 1 | public class Emp { |
| · 2 | private String name; |
| · 3 | public Emp(String name) { |
| · 4 | this.name = name; |
| · 5 | } |
| · 6 | public int hashCode() { |
| · 7 | return 520; |
| · 8 | } |
| · 9 | } |

· 下列说法正确的是：（）。

· A.

由于没有覆盖equals方法，Emp类编译失败

B.

从HashMap中，删除一个Emp类型Key，那么HashMap中的所有对象将被删除

C.

向HashMap中放入已存在的name字符序列的Emp对象，将更改对应的value

D.

HashMap对象可以包含多个name字符序列相同的Emp对象作为Key

· 15.

· (单选题)下列代码的输出结果是：（）。

public static void main(String[] args) {

String str = "LEA";

Deque<Character> stack = new LinkedList<Character>();

for (int i = 0; i < str.length(); i++) {

stack.push(str.charAt(i));

}

stack.pop();

stack.pop();

stack.pop();

System.out.println(stack.peek());

}

· A.

L

B.

E

C.

A

D.

null

· 16.

· (单选题)请看下列代码的输出结果是：（）。

public static void main(String[] args) {

List<Integer> list = new ArrayList<Integer>();

for (int i = 0; i < 10; i++) {

list.add(i);

}

List<Integer> subList = list.subList(4, 9);

for (int i = 0; i < subList.size(); i++) {

subList.set(i, subList.get(i) \* 10);

}

System.out.println(list);

}

· A.

[0, 1, 2, 3, 40, 50, 60, 70, 80, 90]

B.

[0, 1, 2, 3, 4, 50, 60, 70, 80, 90]

C.

[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

D.

[0, 1, 2, 3, 40, 50, 60, 70, 80, 9]

· 17.

· (单选题)现有指向ArrayList对象的集合引用list，该集合的元素为[10,30,50],下列代码用于对集合list进行循环遍历：

|  |  |
| --- | --- |
| · 1 | for(int i=0;i<《插入代码1》;i++){ |
| · 2 | Integer obj=(Integer)《插入代码2》; |
| · 3 | System.out.println(obj); |
| · 4 | } |

· 那么《插入代码1》和《插入代码2》处的代码分别为:（）。

· A.

list.getObject(i) list.length()

B.

list.length() list.getObject(i)

C.

list.size() list.get(i)

D.

list.get(i) list.size()

· 18.

· (单选题)请看下列代码：

public class Grade implements Comparable<Grade> {

private int wins;

private int losses;

public Grade(int w, int l) {

wins = w;

losses = 1;

}

public int getWins() {

return wins;

}

public int getLosses() {

return losses;

}

}完成这个类，需要添加的方法是：（）。

· A.

|  |  |
| --- | --- |
| 1 | public int compareTo(Object o) { |

|  |  |
| --- | --- |
| 2 | return 0; |

|  |  |
| --- | --- |
| 3 | } |

B.

|  |  |
| --- | --- |
| 1 | public int compareTo(Grade o) { |

|  |  |
| --- | --- |
| 2 | return 0; |

|  |  |
| --- | --- |
| 3 | } |

C.

|  |  |
| --- | --- |
| 1 | public int compare(Grade s1,Grade s2){ |

|  |  |
| --- | --- |
| 2 | return 0; |

|  |  |
| --- | --- |
| 3 | } |

D.

|  |  |
| --- | --- |
| 1 | public int compare(Object o1,Object o2){ |

|  |  |
| --- | --- |
| 2 | return 0; |

|  |  |
| --- | --- |
| 3 | } |

· 19.

· (多选题)能与正则表达式http(s)+匹配的是：（）。

· A.

http

B.

https

C.

httpss

D.

s

· 20.

· (多选题)请看下列代码，关于运行后的输出结果说法正确的是:（）。

|  |  |
| --- | --- |
| · 1 | package util; |
| · 2 | class Point {} |
| · 3 | public class Test{ |
| · 4 | public static void main(String[] args) { |
| · 5 | Point p=new Point(); |
| · 6 | System.out.println(p);  System.out.println(p.toString()); |
| · 7 | } |
| · 8 | } |

· A.

输出结果相同

B.

输出结果不同

C.

输出结果都是util.Point@hashCode

D.

其中一个输出结果是Point@hashCode