

CS124 JAVA PROGRAMMING
MULTIPLE CHOICE TEST
SEPTEMBER 1, 2003

Student Name _____
Student Number _____
User-id _____

Please write your solutions on this page, by *circling* your answer. If you change your mind several times, and make it illegible, then use the box at the right to just enter the correct letter.

1	a	b	c	d	e	
2	a	b	c	d	e	
3	a	b	c	d	e	
4	a	b	c	d	e	
5	a	b	c	d	e	
6	a	b	c	d	e	
7	a	b	c	d	e	
8	a	b	c	d	e	
9	a	b	c	d	e	
10	a	b	c	d	e	
11	a	b	c	d	e	
12	a	b	c	d	e	
13	a	b	c	d	e	
14	a	b	c	d	e	
15	a	b	c	d	e	
16	a	b	c	d	e	
17	a	b	c	d	e	
18	a	b	c	d	e	
19	a	b	c	d	e	
20	a	b	c	d	e	

(C) This page intentionally left blank

1. Consider the following instance variable declaration

```
java.awt.Color newColour;
```

This declaration

- (a) has created a variable called `newColour` with initial value 0.
 - (b) has created a variable called `newColour` with an unspecified initial value.
 - (c) will cause a syntax error, as it is not permitted to declare a variable of reference type without creating the associated object.
 - (d) has created a variable called `newColour` referring to a newly created `java.awt.Color` object which by default represents the colour black.
 - (e) (*) has created a variable called `newColour` with initial value `null`.
2. Which of the following methods returns `true` when *exactly one* of the three arguments is `true`, and returns `false` in all other cases

- (a) (*)

```
boolean exactlyOne(boolean a, boolean b, boolean c) {  
    return a && !b && !c || !a && b && !c || !a && !b && c;  
}
```

- (b)

```
boolean exactlyOne(boolean a, boolean b, boolean c) {  
    return (a || b) && (b || c) && (c || a);  
}
```

- (c)

```
boolean exactlyOne(boolean a, boolean b, boolean c) {  
    return !(a && b) || !(b && c) || !(c && a);  
}
```

- (d)

```
boolean exactlyOne(boolean a, boolean b, boolean c) {  
    return a || b || c;  
}
```

- (e)

```
boolean exactlyOne(boolean a, boolean b, boolean c) {  
    return (a || b || c) && !(a && b && c);  
}
```

3. Consider the following declarations

```
java.awt.Color red = new java.awt.Color(255,0,0);  
java.awt.Color xxx = new java.awt.Color(255,0,0);
```

What is the value of the expression `red == xxx`, and why?

- (a) We cannot tell, because it depends on details of the implementation that are hidden from the users.
 - (b) A runtime error occurs because `==` can only be used for primitive types.
 - (c) (*) **false** because they are references to different objects, even though the objects happen to contain identical data.
 - (d) **true** because they both represent the colour red.
 - (e) A reference with the same value as `xxx` because an assignment expression has the same value as the expression on the right hand side of the assignment.
4. What will the value of `a[7]` be after the following statements?

```
int[] a;  
a = new int[8];  
  
a[0] = 1;  
a[1] = 1;  
for (int i=2;i<8;i++) {  
    a[i] = a[i-1]+a[i-2];  
}
```

- (a) 13
- (b) (*) 21
- (c) 34
- (d) 0
- (e) 8

5. What will the method call `mystery(4733)` return, if the method is defined as follows:

```
public int mystery(int n) {  
  
    int m = 0;  
  
    while (n > 0) {  
        m = 10*m + n%10;  
        n = n/10;  
    }  
  
    return m;  
  
}
```

- (a) (*) 3374
 - (b) 17
 - (c) 47330
 - (d) 4733
 - (e) 0
6. What are the values of `a` and `b` *after* the following loop?

```
int a = 1;  
int b = 0;  
  
for (a = 0; b < 10 || a < 20; a = a + 2) {  
    a = a + b;  
    b = b + a + 1;  
}
```

- (a) `a` is 10, and `b` is 16
- (b) `a` is 28, and `b` is 45
- (c) `a` is 12, and `b` is 16
- (d) (*) `a` is 30, and `b` is 45
- (e) `a` is 54, and `b` is 53

The next six questions refer to the following source code, which defines a class that represents circles.

```
public class Circle
{

    // Represents a circle with centre
    // at the point (centreX, centreY)
    // and with the specified radius

    private double centreX;
    private double centreY;
    private double radius;

    public Circle(double x, double y, double r) {
        centreX = x;
        centreY = y;
        radius = r;
    }

    public Circle(double r) {
        this(0,0,r);
    }

    public Circle() {
        this(1d);
    }

    // area() returns the area of the circle

    public double area() {
        return Math.PI * radius * radius;
    }

    // scale() enlarges/shrinks this circle

    public void scale(double factor) {
        radius = radius * factor;
    }
}
```

```

// biggest() returns the circle with the largest area

public Circle biggest(Circle other) {
    if (this.area() > other.area())
        return this;
    else
        return other;
}

// duplicate() returns another circle identical to this one

public Circle duplicate() {
    return new Circle(centreX, centreY, radius);
}

// the mystery method does something unclear

public boolean mystery(double x, double y) {
    double z = (x-centreX)*(x-centreX);
    z = z + (y-centreY)*(y-centreY);
    return z > radius*radius;
}

}

```

7. What are the instance variables of this class?

- (a) x, y and r
- (b) area, scale, biggest and mystery
- (c) area and z
- (d) (*) centreX, centreY and radius
- (e) x, y, r, factor and other

8. Which of the following statements will correctly construct a `Circle`?

1. `Circle c = Circle(5);`
2. `Circle c = new Circle(1,2,1);`
3. `Circle c = new Circle();`
4. `Circle c = Circle(1,2,1).new();`

- (a) 3 only
- (b) (*) 2 and 3 only
- (c) 4 only
- (d) 1, 2 and 3 only
- (e) 2 only

9. During the following sequence of statements, how many `Circle` objects in total are constructed?

```
Circle c1;  
Circle c2;  
Circle c3;  
Circle c4;  
c1 = new Circle(1);  
c2 = c1;  
c3 = c1.duplicate();  
c4 = c3.biggest(c1);
```

- (a) 1
- (b) (*) 2
- (c) 3
- (d) 4
- (e) 5

10. What does the `mystery` method do?
- (a) It tests if the area of the circle is greater than the length of the line joining $(0,0)$ to (x,y) .
 - (b) It returns the radius of the smallest circle containing both (x,y) and $(\text{centreX},\text{centreY})$.
 - (c) It tests if the point (x,y) is exactly on the edge of the circle.
 - (d) It returns the distance between the circle and the point (x,y) .
 - (e) (*) It tests if the point (x,y) is outside the circle.

11. What value is stored in `c_area` after the following code?

```
double c_area;  
Circle c = new Circle(2);  
c.scale(2);  
c_area = c.area();
```

- (a) (*) 50.26548245743669
 - (b) 12.566370614359172
 - (c) 16.0
 - (d) 4.0
 - (e) 3.141592653589793
12. Suppose that `c1` represents a circle with centre $(2,2)$ and radius 1, while `c2` represents a circle with centre $(1,1)$ and radius 2. What is the value of the expression

```
c1.biggest(c2)
```

- (a) An object reference equal to `c1`.
- (b) A reference to a new `Circle` object representing a circle with centre $(1,1)$ and radius 2.
- (c) A reference to a new `Circle` object representing a circle with centre $(2,2)$ and radius 1.
- (d) (*) An object reference equal to `c2`.
- (e) A reference to a new `Circle` object representing a circle with centre $(2,2)$ and radius 2.

13. Suppose that the class `FlagDrawer` contains the following method, where `SimpleCanvas` is defined as in lectures:

```
public void mysteryFlag() {  
  
    SimpleCanvas sc = new SimpleCanvas("Flag",300,300);  
  
    sc.setForegroundColour(new java.awt.Color(0,255,0));  
    for (int i=0;i<300;i++) {  
        sc.drawLine(0,0,i,299-i);  
    }  
  
    sc.setForegroundColour(new java.awt.Color(0,0,255));  
    for (int i=0;i<300;i++) {  
        sc.drawLine(i,299-i,299,299);  
    }  
  
}
```

If you call this method what will be drawn on the screen?

- (a) A square split diagonally into two coloured triangles – a blue one in the top-right half and a green one in the bottom-left half.
- (b) A square split diagonally into two coloured triangles – a red one in the top-right half and a green one in the bottom-left half.
- (c) A square split vertically into two coloured rectangles – a green one in the left-half and a red one in the right half.
- (d) (*) A square split diagonally into two coloured triangles – a green one in the top-left half and a blue one in the bottom-right half.
- (e) A square split horizontally into two coloured rectangles – a blue one in the top half, and a green one in the bottom half.

14. Which one of the following statements is true?
- (a) The source code for a class must explicitly define at least one constructor.
 - (b) The source code for a class can explicitly define at most one constructor.
 - (c) The source code for a class should never contain any explicitly defined constructors.
 - (d) (*) The source code for a class can explicitly define zero, one or more constructors.
 - (e) The source code for a class must explicitly define exactly one constructor.
15. Consider the following class with a single method

```
public class Doubler {  
    public void doubleIt(int n) {  
        n = 2*n;  
    }  
}
```

What happens when the class is compiled, and the following sequence of statements (which is in a method of another class) is compiled and executed?

```
int x = 20;  
Doubler d = new Doubler();  
d.doubleIt(x);  
System.out.println(x);
```

- (a) The code for `Doubler` will not compile because there is no instance variable called `n`.
- (b) The value 40 will be printed out on the terminal window.
- (c) (*) The value 20 will be printed out on the terminal window.
- (d) The code for `Doubler` will not compile, because the class has no constructors.
- (e) The code for `Doubler` will not compile because a parameter variable is read-only and cannot be assigned a value.

16. Consider a method in the class `BankAccount` (defined as in lectures) with the signature

```
public boolean higherBalance(BankAccount other)
```

that should test whether the target object has a strictly higher balance than the argument object (*strictly* means that the method should return `false` if the balances are equal).

Three methods are proposed for this purpose:

```
public boolean higherBalance(BankAccount other) {  
    return balance > other.balance;  
}
```

```
public boolean higherBalance(BankAccount other) {  
    if (getBalance() > other.getBalance()) {  
        return this;  
    } else {  
        return other;  
    }  
}
```

```
public boolean higherBalance(BankAccount other) {  
    if (balance <= other.getBalance()) {  
        return false;  
    } else {  
        return true;  
    }  
}
```

Which of them will work correctly?

- (a) (*) The 1st and 3rd only
- (b) The 2nd and 3rd only
- (c) The 3rd only
- (d) All three of them
- (e) The 1st and 2nd only

17. What are the values of the following three expressions respectively?

```
12 / 8 * 3
12 / 8d * 3
12 / 8 * (double)3
```

- (a) 3, 3.0 and 3.0
- (b) 3.0, 3.0 and 4.5
- (c) (*) 3, 4.5 and 3.0
- (d) 4, 4.5 and 3.0
- (e) 4.5, 4.5 and 4.5

18. Suppose that a variable `int runTime` contains the running time of a movie in minutes, and that you wish to convert it into the normal hours-and-minutes notation (for example, if the movie is 133 minutes long, then it is 2 hours and 13 minutes long).

Which of the following pieces of code will calculate the correct numbers of hours and minutes?

- (a)

```
int minutes = runTime%60;
int hours = runTime-60*minutes;
```
- (b) (*)

```
int minutes = runTime%60;
int hours = runTime/60;
```
- (c)

```
int hours = runTime-60;
int minutes = runTime/60;
```
- (d)

```
int hours = runTime/60;
int minutes = (runTime - hours*60)/60;
```
- (e)

```
int hours = runTime%60;
int minutes = runTime;
```

19. Assume `BankAccount` is defined as in lectures, and consider the following sequence of statements.

```
BankAccount b1 = new BankAccount("gfr",1,0);
BankAccount b2 = new BankAccount("rlw",2,0);
BankAccount temp = b1;
b1.deposit(1000);
b2.deposit(2000);
temp.withdraw(500);
temp = b2;
temp.deposit(1000);
```

What is the balance in the accounts owned by “gfr” and “rlw” respectively?

- (a) 1000 and 3000
 - (b) 500 and 2500
 - (c) 1500 and 2000
 - (d) (*) 500 and 3000
 - (e) 1000 and 2000
20. The *alternating sum* of an array `a` is equal to the value

$$a[0] - a[1] + a[2] - a[3] + \dots$$

Which of the following three methods correctly calculates the alternating sum of the argument array?

```
// Method 1
public int alternatingSum(int[] a) {
    int sum = 0;
    for (int i=0; i<a.length; i=i+2) {
        sum = sum + a[i] - a[i+1];
    }
    return sum;
}
```

```

// Method 2
public int alternatingSum(int[] a) {
    int sum = 0;
    for (int i=0; i<a.length; i++) {
        if (i % 2 == 0) {
            sum = sum - a[i];
        } else {
            sum = sum + a[i];
        }
    }
    return sum;
}

```

```

// Method 3
public int alternatingSum(int[] a) {
    int sum = 0;
    for (int i=0; i<a.length; i++) {
        sum = a[i] - sum;
    }
    if (a.length % 2 == 1) {
        return sum;
    } else {
        return -sum;
    }
}

```

- (a) (*) Method 3 only
- (b) Methods 1 and 3 only
- (c) None of them
- (d) Method 1 only
- (e) Method 2 only

BLANK PAGE FOR ROUGH WORK

BLANK PAGE FOR ROUGH WORK

BLANK PAGE FOR ROUGH WORK