(1)

```
class Base {
    // insert code here
}

public class Derived extends Base {

    public static void main(String[] args) {
        Derived obj = new Derived();
        obj.setNum(3);
        System.out.println("Square = " + obj.getNum() * obj.getNum());
    }
}
```

A(ans)

```
private int num;

public int getNum() {
    return num;
}

public void setNum(int num) {
    this.num = num;
}
```

B

```
public int num;

protected public int getNum() {
    return num;
}

protected public void setNum(int num) {
    this.num = num;
}
```

C

```
private int num;

public int getNum() {
    return num;
}

private void setNum(int num) {
    this.num = num;
}
```

D(ans)

```
protected int num;

public int getNum() {
    return num;
}

public void setNum(int num) {
    this.num = num;
```

```
}

E
protected int num;

private int getNum() {
    return num;
}

public void setNum(int num) {
    this.num = num;
}
```

題解

選項 A，標準的封裝作法。
選項 B，protected 和 public 不可以同時使用，會造成編譯錯誤。
選項 C，應將 setNum 方法的可見度修飾字改為 public。
選項 D，標準的封裝作法，支援不同套件下的欄位繼承。
選項 E，應將 getNum 方法的可見度修飾字改為 private。

(2)

```
class Overloading {

    int x(double d) {
        System.out.println("one");
        return 0;
    }

    String x(double d) {
        System.out.println("two");
        return null;
    }

    double x(double d) {
        System.out.println("three");
        return 0.0;
    }

    public static void main(String[] args) {
        new Overloading().x(4.0);
    }
}

What is the result?

A
One

B
Two

C
Three

D(ans)
```

Compilation fails.

題解

程式第 8 行和第 13 行，x 方法的簽名(Signature)都是相同的，因此會發生編譯錯誤。

(3)

```
public class Whizlabs {

    public static void main(String[] args) {
        String s = "A";

        switch (s) {
            case "a":
                System.out.println("simaple A ");
            default:
                System.out.print("default ");
            case "A":
                System.out.print("Capital A ");
        }
    }
}
```

What is the result?

A
simaple A

B(ans)
Capital A

C
simaple A default Capital A

D
simaple A default

E
Compilation fails.

題解

switch 要選擇的內容是 s 變數所參考到的「A」字串，因此會執行第 11 行的 case，輸出「Capital A 」後就離開 switch 了。

(4)

Given the following class:

```
public class CheckingAccount {

    public int amount;

    public CheckingAccount(int amount) {
        this.amount = amount;
    }

    public int getAmount() {
        return amount;
```

```
    }

    public void changeAmount(int x) {
        amount += x;
    }
}


public static void main(String[] args){
    CheckingAccount acct = new CheckingAccount((int)(Math.random() *
1000));
    //line n1
    System.out.println(acct.getAmount());
}
```

Which three lines, when inserted independently at line n1, cause the program to print a 0 balance?

A
```
this.amount = 0
```

B
```
amount = 0;;
```

C
```
acct(0);
```

D(ans)
```
acct.amount = 0;
```

E
```
acct.getAmount() = 0;
```

F
```
acct.changeAmount(0);
```

G(ans)
```
acct.changeAmount(-acct.amount);
```

H(ans)
```
acct.changeAmount(-acct.getAmount());
```

題解

選項 A，由於 main 方法是在不同的類別，因此使用「this」並不能存取到 acct 物件，會造成編譯錯誤。
選項 B，amount 因為沒有宣告，所以會造成編譯錯誤。
選項 C，acct 是物件變數，不是方法，無法這樣使用。
選項 D，直接存取 acct 變數參考到的物件之欄位，是可行的作法。
選項 E，方法不能這樣使用，會造成編譯錯誤。
選項 F，因為無法保證 amount 一開始是 0，所以將它加 0 並不一定可以輸出 0。
選項 G，amount = amount - amount，所以結果會是 0。
選項 H，理由同選項 G。

(5)
```
public class Test {
    static String[][] arr = new String[3][];
```

```
    private static void doPrint() {
        //insert code here
    }

    public static void main(String[] args) {
        String[] class1 = {"A", "B", "C"};
        String[] class2 = {"L", "M", "N", "O"};
        String[] class3 = {"I", "J"};
        arr[0] = class1;
        arr[1] = class2;
        arr[2] = class3;
        Test.doPrint();
    }
}
```

Which code fragment, when inserted at line //insert code here, enables the code to print COJ?

A
```
int i = 0;
for (String[] sub : arr) {
    int j = sub.length - 1;
    for (String str : sub) {
        System.out.println(str[j]);
        i++;
    }
}
```

B(ans)
```
for (int i = 0; i < arr.length; i++) {
    int j = arr[i].length - 1;
    System.out.print(arr[i][j]);
}
```

C
```
int i = 0;
for (String[] sub : arr[][]) {
    int j = sub.length;
    System.out.print(arr[i][j]);
    i++;
}
```

D
```
for (int i = 0; i < arr.length - 1; i++) {
    int j = arr[i].length - 1;
    System.out.print(arr[i][j]);
    i++;
}
```

題解

選項 A，第 10 行會編譯錯誤，因為 str 變數的型態不是陣列。
選項 B，輸出二維陣列每個一維陣列元素的最後一個元素，正好是「COJ」。
選項 C，第 7 行會編譯錯誤，錯誤的 foreach 用法。
選項 D，只會輸出「C」。

(6)

Given the code fragment:

```
public static void main(String[] args) {
    int ii = 0;
    int jj = 7;
    for (ii = 0; ii < jj - 1; ii = ii + 2) {
        System.out.print(ii + " ");
    }
}
```

What is the result?

A
2 4

B
0 2 4 6

C(ans)
0 2 4

D
Compilation fails

題解

程式第 5 行的 for 迴圈，第一次執行時 ii = 0，ii < 6，所以迴圈執行，輸出「0」；第一次執行時 ii = 2，ii < 6，所以迴圈執行，輸出「2」；第一次執行時 ii = 4，ii < 6，所以迴圈執行，輸出「4」；第三次執行時 ii = 6，ii = 6，所以立刻跳出迴圈。

(7)

```
public class MyFor3 {
    public static void main(String[] args) {
        int[] xx = null;
        for (int ii : xx) {
            System.out.println(ii);
        }
    }
}
```

A
null

B
Compilation fails

C(ans)
An exception is thrown at runtime

D
0

題解

程式會在第 5 行拋出 NullPointerException，因為 xx 變數的儲存的陣列物件參考是 null。

(8)

```
public class TestApp {
    public static void main(String[] args) {
        TestApp t = new TestApp();
```

```
        try {
            t.doPrint();
            t.doList();
        } catch (Exception e2) {
            System.out.println("Caught " + e2);
        }
    }
    public void doList() throws Exception {
        throw new Error("Error");
    }
    public void doPrint() throws Exception {
        throw new RuntimeException("Exception");
    }
}
```

A
```
Caught java.lang.RuntimeException: Exception
Exception in thread "main" java.lang.Error: Error
at TestApp.doList(TestApp.java: 14)
at TestApp.main(TestApp.java: 6)
```

B
```
Exception in thread "main" java.lang.Error: Error
at TestApp.doList(TestApp.java: 14)
at TestApp.main(TestApp.java: 6
```

C
```
Caught java.lang.RuntimeException: Exception
Caught java.lang.Error: Error
```

D(ans)
Caught java.lang.RuntimeException: Exception
題解
程式第 5 行會呼叫第 14 行的 doPrint 方法，然後拋出一個 RuntimeException 物件。這個 RuntimeException 會在程式第 7 行的 catch 被接住，然後輸出「Caught java.lang.RuntimeException: Exception」。

(9)
Which of the following will print current time?

A
```
System.out.print(new LocalTime().now());
```

B
```
System.out.print(new LocalTime());
```

C(ans)
System.out.print(LocalTime.now());

D
```
System.out.print(LocalTime.today());
```

E
None of the above.
題解

這題只是在測驗 LocalTime 類別的用法

(10)

```
public class Test {

    public static void main(String[] args) {
        if (args[0].equals("Hello") ? false : true) {
            System.out.println("Success");
        } else {
            System.out.println("Failure");
        }
    }
}
```

```
javac Test.Java
java Test Hello
```

A
Success

B(ans)
Failure

C
Compilation fails.

D
An exception is thrown at runtime

題解

題目執行的指令會編譯「Test.java」，由於 Test 類別有用 public 修飾，因此預設會執行 Test 類別的 main 方法，並把「Hello」作為 main 的參數。參數的索引值從 0 開始。「args[0].equals("Hello") ? false : true」中的「args[0].equals("Hello")」會判斷 args[0]所參考到的字串物件是否在邏輯上和「Hello」字串物件相同，由於第一個(索引值 0)參數輸入「Hello」，因此這個的結果會是 true。再來這個結果會代入三元條件的運算元中，所以會變成「true ? false : true」，將回傳 false。最後程式輸出「Failure」。

(11)

```
class Cake {
    int model;
    String flavor;

    Cake() {
        model = 0;
        flavor = "Unknown";
    }
}

public class Test {
    public static void main(String[] args) {
        Cake c = new Cake();
        bake1(c);
        System.out.println(c.model + " " + c.flavor);
        bake2(c);
        System.out.println(c.model + " " + c.flavor);
    }
```

```
    public static Cake bake1(Cake c) {
        c.flavor = "Strawberry";
        c.model = 1200;
        return c;
    }

    public static void bake2(Cake c) {
        c.flavor = "Chocolate";
        c.model = 1230;
        return;
    }
}
```

A
```
0 unknown
0 unknown
```

B
```
1200 Strawberry
1200 Strawberry
```

C(ans)
```
1200 Strawberry
1230 Chocolate
```

D
```
Compilation fails
```

題解

Cake 類別和其物件欄位都沒有加上修飾字，因此預設的存取範圍在同一個套件之下，當然也包括第 12 行的 Test 類別。

程式第 22 行的 bake1 方法，將物件欄位設定為 c.flavor = "Strawberry", c.model = 1200。

程式第 28 行的 bake2 方法，將物件欄位設定為 c.flavor = "Chocolate", c.model = 1230。

(12)
```
int i, j = 0;
i = (3 * 2 + 4 + 5);
j = (3 * ((2 + 4) + 5));
System.out.println("i:" + i + "\nj:" + j);
```

A
```
i:16
j:33
```

B(ans)
```
i.15
j:33
```

C
```
i:33
j:23
```

D
i:15
j:23
題解
Java 的算式遵循「先乘除，後加減」和「括號內先計算」的規則，因此此題的運算過程如下：
i = (3 * 2 + 4 + 5) = (**6** + 4 + 5) = (**10** + 5) = 15
j = (3 * ((2 + 4) + 5)) = (3 * (**6** + 5)) = (3 * **11**) = 33

(13)
Which two actions will improve the encapsulation of a class?

A(ans)
Changing the access modifier of a field from public to private

B
Removing the public modifier from a class declaration

C
Changing the return type of a method to void

D(ans)

Returning a copy of the contents of an array or ArrayList instead of a direct reference
題解
選項 A，把欄位用 private 修飾字來宣告成私有成員，接著再使用 setter 和 getter 方法來提供給外部存取確實會比較符合封裝的概念。
選項 B，為了要讓類別可以被外部使用，還是需要公開的方法的。
選項 C，也要有可以回傳數值或是物件的方法，不然類別的作用不大。
選項 D，複製一份新的陣列物件可以避免封裝內的陣列物件裡的內容在外部被改變。

(14)
Given the code fragment:
```
float x = 22.00f % 3.00f;
int y = 22 % 3;
System.out.print(x + ", " + y);
```

A(ans)
1.0, 1

B

1.0f, 1

C
7.33, 7

D
Compilation fails

E
An exception is thrown at runtime

題解
「%」是餘數運算子，支援整數和單倍精準浮點數(float)型態的餘數計算。若運算元的其中一個為單倍精準浮點數型態，則計算出來的結果也會是單倍精準浮點數型態。
22 除以 3 的結果為 7 餘 1，因此第 6 行會算出單倍精準浮點數型態的「1」，而第 7 行會算出整數型態的「1」。

(15)

```
public class Equal {
    public static void main(String[] args) {
        String str1 = "Java";
        String[] str2 = {"J", "a", "v", "a"};
        String str3 = "";
        for (String str : str2) {
            str3 = str3 + str;
        }
        boolean b1 = (str1 == str3);
        boolean b2 = (str1.equals(str3));
        System.out.print(b1 + ", " + b2);
    }
}
```

A
true, false

B(ans)
false, true

C
true, true

D
false, false
題解
程式執行到第 10 行時，str1 和 str3 變數所參考到的都是「Java」字串物件，只是是不同的字串物件。因此第 10 行判斷參考是否相同的「==」運算子，會得到 false 的結果。而第 11 行判斷邏輯是否相同的「equals」方法會得到 true 的結果。

(16)

```
public class MyClass {
    public static void main(String[] args) {
        String s = " Java Duke ";
        int len = s.trim().length();
        System.out.print(len);
    }
}
```

A
8

B(ans)
9

C
11

D
10

E
Compilation fails

題解

String 物件的 trim 方法可以消除開頭和結尾的所有空格，回傳新的字串，並不會改變原有字串物件的內容。所以在這題中，第 5 行會將「 Java Duke 」字串去除掉頭尾的空格後，變成「Java Duke」，所以字串長度會變成「9」。

(17)

The protected modifier on a Field declaration within a public class means that the field _____.

A
Cannot be modified

B
Can be read but not written from outside the class

C
Can be read and written from this class and its subclasses only within the same package

D(ans)

Can be read and written from this class and its subclasses defined in any package

題解

protected 修飾子可以使成員「能在同一個套件內自由存取，不同的套件需要有繼承關係才能存取」。

(18)

```
class Student {
    int rollnumber;
    String name;
    List cources = new ArrayList();
        // insert code here
    public String toString() {
        return rollnumber + " : " + name + " : " + cources;
    }
}

public class Test {
    public static void main(String[] args) {
        List cs = new ArrayList();
        cs.add("Java");
        cs.add("C");
        Student s = new Student(123, "Fred", cs);
        System.out.println(s);
    }
}
```

Which code fragment, when inserted at line // insert code here, enables class Test to print 123 : Fred : [Java, C]?

A
```
private Student(int i, String name, List cs) {
/* initialization code goes here */
}
```

B
```
public void Student(int i, String name, List cs) {
/* initialization code goes here */
}
```

C(ans)
```
Student(int i, String name, List cs) {
/* initialization code goes here */
}
```

D
```
Student(int i, String name, ArrayList cs) {
/* initialization code goes here */
}
```

題解

題目原先的程式會在第 26 行發生編譯錯誤，原因在於 Student 類別內並沒有實作出 Student(int i, String name, List cs)這樣的建構子。因此必須要加入建構子至程式第 9 行。

選項 A，建構子若使用 private 來修飾，就無法在類別之外被實體化了。

選項 B，建構子不能撰寫回傳值的型態，連無回傳值型態「void」都不行。

選項 C，正確答案。

選項 D，因為隱含式(implicit)的自動轉型只能向上轉型，因此第三個參數應該要是 List 型態或是 List 繼承之父類別的物件型態(如 Collection、Iterable、Object)，而不能是繼承 List 的子類別(如 ArrayList、LinkedList)。

(19)
```
public class Circle {
    double radius;
    public double area;

    public Circle(double r) {
        radius = r;
    }
    public double getRadius() {
        return radius;
    }
    public void setRadius(double r) {
        radius = r;
    }
    public double getArea() {
        return /* ??? */;
    }
}

class App {
    public static void main(String[] args) {
        Circle c1 = new Circle(17.4);
        c1.area = Math.PI * c1.getRadius() * c1.getRadius();
    }
}
```

Which two modifications are necessary to ensure that the class is being properly encapsulated?

A(ans)
Remove the area field

B(ans)
Change the getArea( ) method as follows:
public double getArea ( ) { return Match.PI * radius * radius; }

C
Add the following method:
public double getArea ( ) {area = Match.PI * radius * radius; }

D
Change the cacess modifier of the SerRadius ( ) method to be protected.
題解
為了讓 Circle 類別計算面積的結果不被干擾，我們可以單純使用 getArea 方法來回傳結果值。所以先移除掉 area 欄位，接著在 getArea 方法實作面積的算法(πr^2)。所以答案是選項 A 和選項 B。
選項 C，會造成編譯錯誤，因為方法有定義回傳型態，卻沒有回傳任何數值。
選項 D，將 setRadius 改成使用 protected 修飾，只會讓它在不同的套件下無法被存取。

(20)
```
public class Series {

    public static void main() {
        int arr[] = {1, 2, 3};
        for (int var : arr) {
            int i = 1;
            while (i <= var);
            System.out.println(i++);
        }
    }
}
```
What is the result?

A
1
1
1

B
1
2
3

C
2
3
4

D

Compilation fails

E(ans)
The loop executes infinite times
題解
這裡要注意到程式第 7 行的 while 迴圈最後加了分號，因此 while 迴圈的範圍只在第 7 行。「i <= var」永遠都會成立，變成無窮迴圈。

(21)
Given the following class declarations:
public abstract class Animal
public interface Hunter
public class Cat extends Animal implements Hunter
public class Tiger extends Cat
Which answer fails to compile?

A
```
ArrayList<Animal> myList = new ArrayList<>();
myList.add(new Tiger());
```

B
```
ArrayList<Hunter> myList = new ArrayList<>();
myList.add(new Cat());
```

C
```
ArrayList<Hunter> myList = new ArrayList<>();
myList.add(new Tiger());
```

D(ans)
```
ArrayList<Tiger> myList = new ArrayList<>();
myList.add(new Cat());
```

E
```
ArrayList<Animal> myList = new ArrayList<>();
myList.add(new Cat());
```
題解
選項 A，Tiger 是 Animal(Tiger 繼承 Cat，Cat 繼承 Animal)，所以這沒有問題。
選項 B，Cat 是 Hunter(Cat 繼承 Animal)，所以這沒有問題。
選項 C，Tiger 是 Hunter(Tiger 繼承 Cat，Cat 實作 Hunter)，所以這沒有問題。
選項 D，Cat 不是 Tiger，會編譯錯誤。
選項 E，Cat 不是 Animal(Cat 繼承 Animal)，所以這沒有問題。

(22)
Given the code fragment:
```
// insert code here
arr[0] = new int[3];
arr[0][0] = 1;
arr[0][1] = 2;
arr[0][2] = 3;

arr[1] = new int[4];
arr[1][0] = 10;
arr[1][1] = 20;
arr[1][2] = 30;
arr[1][3] = 40;
```

Which two statements, when inserted independently at line // insert code here, enable the code to compile?

A
```
int[][] arr = null;
```

B
```
int[][] arr = new int[2];
```

C(ans)
```
int[][] arr = new int[2][];
```

D
```
int[][] arr = new int[][4];
```

E(ans)
```
int[][] arr = new int[2][4];
```

F
```
int[][] arr = new int[0][4];
```

題解
從程式第 6 到 15 行，可以斷定這 arr 變數參考到的陣列大小至少為 2 x 4。
選項 A，沒有實體化二維陣列物件，會在之後存取陣列元素時拋出 NullPointerException。
選項 B，產生一維陣列的物件，使用二維陣列的型態來儲存會發生編譯錯誤。
選項 C，正確的二維陣列實體化方式，大小也符合需求
選項 D，錯誤的二維陣列實體化方式。
選項 E，正確的二維陣列實體化方式，大小也符合需求。
選項 F，正確的二維陣列實體化方式，但是大小不符合需求。

(23)
Given the for loop construct:
```
for ( expr1 ; expr2 ; expr3 ) {
    statement;
}
```

Which two statements are true?

A(ans)
This is not the only valid for loop construct; there exits another form of for loop constructor.

B(ans)
The expression expr1 is optional. it initializes the loop and is evaluated once, as the loop begin.

C
When expr2 evaluates to false, the loop terminates. It is evaluated only after each iteration through the loop.

D
The expression expr3 must be present. It is evaluated after each iteration through the loop.

題解
選項 A，敘述正確，還有 foreach 的結構，如下：

```
for ( expr1 : expr2) {
    statement;
}
```

選項 B，敘述正確，expr1 是用來初始化 for 迴圈會用到的變數。
選項 C，每次要進入迴圈之前就會判斷 expr2，而不是每次迴圈執行之後。每次迴圈執行之後才判斷 expr2，比較像是 do-while 的結構。
選項 D，expr3 可以省略沒有關係。

(24)

```
int a[] = {1, 2, 3, 4, 5};
for (XXX) {
    System.out.print(a[e]);
}
```

Which option can replace xxx to enable the code to print 135?

A

```
int e = 0; e <= 4; e++
```

B(ans)

```
int e = 0; e < 5; e += 2
```

C

```
int e = 1; e <= 5; e += 1
```

D

```
int e = 1; e < 5; e +=2
```

題解
選項 A，會輸出「12345」。
選項 B，會輸出「135」，為正確答案。
選項 C，會輸出「1234」，然後因為 a[5] 超出陣列範圍，拋出 ArrayIndexOutOfBoundsException。
選項 D，會輸出「24」。

(25)

```
public class X implements Z {

    public String toString() {
        return "X ";
    }

    public static void main(String[] args) {
        Y myY = new Y();
        X myX = myY;
        Z myZ = myX;
        System.out.print(myX);
        System.out.print((Y) myX);
        System.out.print(myZ);
    }
}
```

```
class Y extends X {
    public String toString() {
        return "Y ";
    }
}

interface Z {
    public String toString();
}
```

What is the output?

A
X X X

B
X Y X

C
Y Y X

D(ans)
Y Y Y
題解
print 方法在傳入物件的時候，會去呼叫物件的 toString 來取得字串值。由於 toString 是物件方法，因此只需注意物件實體是哪個就好，多型型態不用去管。
程式執行完第 10 行後，myY 變數所參考到的物件是 Y 物件，myX 變數所參考到的物件是 Y 物件，myZ 變數所參考到的物件也是 Y 物件，這三個 Y 物件都是同一個物件。第 11 行之後的輸出，會去執行第 19 行 Y 類別內的 toString 物件方法，所以每個都會輸出「Y 」。

(26)
```
public class ComputeSum {
    public int x;
    public int y;
    public int sum;

    public ComputeSum(int nx, int ny) {
        x = nx;
        y = ny;
        updateSum();
    }

    public void setX(int nx) {
        x = nx;
        updateSum();
    }

    public void setY(int ny) {
        x = ny;
        updateSum();
    }

    void updateSum() {
        sum = x + y;
```

```
    }
}
```

This class needs to protect an invariant on the sum field.
Which three members must have the private access modifier to ensure that this invariant is maintained?

A.
The x field

B.
The y field

C. (ans)
The sum field

D.
The ComputerSum ( ) constructor

E. (ans)
The setX ( ) method

F. (ans)
The setY ( ) method

題解
只有在選項 C、D、E、F 內會更動到 sum 欄位的值，但是選項 D 是建構子，在物件實體化的時候才會執行，實體化之就不能再被呼叫了，也就再也沒有機會可以去更動 sum 欄位的值。
因此去掉選項 D 之後，答案便是選項 C、E、F

(27)
```
int[] array = {1, 2, 3, 4, 5};
```
And given the requirements:
Process all the elements of the array in the order of entry.
Process all the elements of the array in the reverse order of entry.
Process alternating elements of the array in the order of entry.
Which two statements are true?

A
Requirements 1, 2, and 3 can be implemented by using the enhanced for loop.

B(ans)
Requirements 1, 2, and 3 can be implemented by using the standard for loop.

C
Requirements 2 and 3 CANNOT be implemented by using the standard for loop.

D(ans)
Requirement 1 can be implemented by using the enhanced for loop.

E
Requirement 3 CANNOT be implemented by using either the enhanced for loop or the standard for loop.

題解

需求 1，按照順序走訪 array 陣列物件可以使用標準的 for loop 或是增強的 for loop(即 foreach)來實作出來。程式如下：

```
for(int i = 0; i < array.length; ++i){
    int number = array[i];
}
for(int number : array){
}
```

需求 2，反向走訪 array 陣列物件只可以使用標準的 for loop 來實作。

```
for(int i = array.length - 1; i >= 0; --i){
    int number = array[i];
}
```

需求 3，使用標準的 for loop，可以間格走訪 array 陣列物件，實作方式如下：

```
for(int i = 0; i < array.length; i += 2){
    int number = array[i];
}
```

所以選項 B 和選項 D 的敘述是正確的。

(28)

```
public class Whizlabs {
    public static void main(String[] args) {
        try {
            Double number = Double.valueOf("120D");
        } catch (NumberFormatException ex) {
        }
        System.out.println(number);
    }
}
```

What is the result?

A
120

B120D

C
A NumberFormatException will be thrown.

D
Compilation fails due to error at line 5.

E(ans)

Compilation fails due to error at line 8.

題解

程式第 5 行，使用 Double 類的 valueOf 方法將字串轉成 Double 物件。

「120D」或是「120d」在 Java 中是正確的數值表示方式，表示為 double 型態的 120。若為「120F」或是「120f」，則表示為 float 型態的 120。若為「120」或是「120」，則表示為 int 型態的 120。程式碼舉例如下：

```
int a = 120;
float b = 120f;
```

```
double c = 120d;
```
因此程式第 5 行會成功執行並產生 Double 物件，而不會拋出例外。但是在程式第 8 行，有用到 number 變數，但 number 變數只有在第 4 行開始的 try 程式區塊中被定義，有效範圍只有在 try 程式區塊中。因此程式第 8 行會因為沒有定義 number，而發生編譯錯誤。

(29)

Which statement is true about the default constructor of a top-level class?

A
It can take arguments.

B
It has private access modifier in its declaration.

C
It can be overloaded.

D(ans)

The default constructor of a subclass always invokes the no-argument constructor of its superclass.

題解

選項 A，預設的建構子不能傳入引數。
選項 B，預設的建構子是以 public 來修飾。
選項 C，建構子不能被覆寫。
選項 D，預設的建構子預設會去呼叫上層無參數的建構子。例如 A 類別的預設建構子，自動產生的程式碼如下：

```
public A(){
    super();
}
```

(30)

Which statement is true about Java byte code?

A
It can run on any platform.

B
It can run on any platform only if it was compiled for that platform.

C(ans)

It can run on any platform that has the Java Runtime Environment.

D
It can run on any platform that has a Java compiler.

E
It can run on any platform only if that platform has both the Java Runtime Environment and a Java compiler.

題解

Java 只能執行在 Java Runtime Environment(JRE)上，換句話說任何平台只要有安裝正確版本的 JRE，都可以執行 Java。

(31)

```
System.out.println(28 + 5 <= 4 + 29);
System.out.println((28 + 5) <= (4 + 29));
```

A
28false29
true

B
285 < 429 true

C(ans)
true
true

D
compilation fails

題解
「+」在其中至少一個運算元為字串(物件)的時候，才會進行字串連接的計算。「<=」是關係運算子，左邊運算元「小於等於」右邊運算元時會回傳布林「true」，否則回傳布林「false」。 因此這題計算方式如下：
28 + 5 <= 4 + 29 = (28 + 5) <= (4 + 29) = 33 <= 33 = true

(32)
```
public class Test {
    public static List data = new ArrayList();

    // insert code here
    {
        for (String x : strs) {
            data.add(x);
        }
        return data;
    }

    public static void main(String[] args) {
        String[] d = {"a", "b", "c"};
        update(d);
        for (String s : d) {
            System.out.print(s + " ");
        }
    }
}
```

Which code fragment, when inserted at // insert code here, enables the code to compile and and print a b c?
A
List update (String[] strs)

B
static ArrayList update(String[] strs)

C(ans)
static List update (String[] strs)

D
static void update (String[] strs)

E
ArrayList static update(String[] strs)

題解

題目原先提供的程式，第 7 行的 strs 會因為沒有宣告而編譯錯誤。第 10 行的 return 會因為目前程式區塊不是可以回傳 List 物件的方法而編譯錯誤。第 15 行會因為找不到 update 方法而編譯錯誤。

為了解決編譯錯誤的問題，必須要在第 5 行完成方法的宣告。方法的名稱叫作「update」，因為要可以直接在 main 方法中使用，所以一定是類別靜態(static)方法，要用 static 修飾。update 方法只能接受字串陣列引數的傳入，所以要設定一個字串陣列的參數，至於參數名稱可以從第 7 行知道，就是「strs」這個變數名稱。第 10 行需要回傳 List 物件，所以 update 方法的回傳值型態應該要宣告成 List 或是 List 的父類別型態。

選項 A、B、D、E 都不符合要求，只有選項 C 是正確的。

(33)

What is the name of the Java concept that uses access modifiers to protect variables and hide them within a class?

A(ans)

Encapsulation

B

Inheritance

C

Abstraction

D

Instantiation

E

Polymorphism

題解

封裝的概念就是保護程式不被修改和隱藏程式實作(程式碼)。

(34)

```
public class Palindrome {
    public static int main(String[] args) {
        System.out.print(args[1]);
        return 0;
    }
}
```

```
javac Palindrome.java
java Palindrome Wow Mom
```

What is the result?

A

Compilation fails

B(ans)

The code compiles, but does not execute.

C

Paildrome

D

Wow

E
Mom

題解

Java 程式進入點的 main 方法宣告方式為「public static void main(String[] args) 」或是「public static void main(String... args) 」。在這裡並沒有宣告出正確的程式進入點，回傳型態不是 void，因此使用「java」指令執行時，只會提示回傳型態必須使用 void。

(35)

```
public class Access {
    private int x = 0;
    private int y = 0;

    public static void main(String[] args) {
        Access accApp = new Access();
        accApp.printThis(1, 2);
        accApp.printThat(3, 4);
    }

    public void printThis(int x, int y) {
        x = x;
        y = y;
        System.out.println("x:" + this.x + " y:" + this.y);
    }

    public void printThat(int x, int y) {
        this.x = x;
        this.y = y;
        System.out.println("x:" + this.x + " y:" + this.y);
    }
}
```

A
x:1 y:2
x:3 y:4

B(ans)
x:0 y:0
x:3 y:4

C
x:1 y:2
x:0 y:0

D
x:0 y:0
x:0 y:0

題解

第 13、14 行，printThis 方法的 x、y 參數會遮蔽 Access 類別的 x、y 物件欄位，所以 Access 物件欄位的值並不會被修改到，維持原來的 x=0、y=0。

第 19、20 行，printThat 方法的 x、y 參數會遮蔽 Access 類別的 x、y 物件欄位，但在這裡使用了「this」來表示目前 Access 物件的參考，所以可以存取到 Access 物件欄位，欄位的值會成功被修改，變成 x=3、y=4。

(36)

```
public static void main(String[] args) {
    int iVar = 100;
    float fVar = 100.100f;
    double dVar = 123;
    iVar = fVar;
    fVar = iVar;
    dVar = fVar;
    fVar = dVar;
    dVar = iVar;
    iVar = dVar;
}
```

Which three lines fail to compile?

A(ans)
Line 7

B
Line 8

C
Line 9

D(ans)
Line 10

E
Line 11

F(ans)
Line 12

題解

int 是 32 位元的整數型態；float 是 32 位元的浮點數型態；double 是 64 位元的浮點數型態。

int 可以隱含式(implicit)轉換成 float 或是 double，float 可以隱含式(implicit)轉換成 double。

所以這題 float 轉 int 的 Line 7、double 轉 float 的 Line 10 和 double 轉 int 的 Line 12 會編譯錯誤。

(37)

```
public class MarkList {
    int num;

    public static void graceMarks(MarkList obj4) {
        obj4.num += 10;
    }

    public static void main(String[] args) {
        MarkList obj1 = new MarkList();
        MarkList obj2 = obj1;
        MarkList obj3 = null;
```

```
        obj2.num = 60;
        graceMarks(obj2);
    }
}
```

How many objects are created in the memory runtime?

A(ans)
1

B
2

C
3

D
4

題解
僅在第 10 行有用 new 運算子實體化出一個 MarkList 物件。

(38)
Given the content of three files:
A.java

```
public class A {
    public void a() {
    }
    int a;
}
```

B.java

```
public class B {
    private int doStuff(){
        private int x = 100;
        return x++;
    }
}
```

C.java

```
import java.io.*;
package p1;

class C {
    public void main(String fileName) throws IOException {
    }
}
```

Which statement is true?

A(ans)
Only the A.java file compiles successfully.

B
Only the B.java file compiles successfully.

C
Only the C.java file compiles successfully.

D
The A.Java and B.java files compile successfully.

E
The B.java and C.java files compile successfully.

F
The A.Java and C.java files compile successfully.

題解

「A.java」檔案可以成功編譯。

「B.java」檔案會編譯失敗，因為 private、protected、public 等可見度修飾字只能使用在類別或是物件的欄位和方法，並不能用在區域變數。

「C.java」檔案會編譯失敗，因為「package」只能在所有程式敘述的最上方。

(39)

What is the proper way to defined a method that take two int values and returns their sum as an int value?

A.
```
int sum(int first, int second) { first + second; }
```

B.
```
int sum(int first, second) { return first + second; }
```

C.
```
sum(int first, int second) { return first + second; }
```

D. (ans)
```
int sum(int first, int second) { return first + second; }
```

E.
```
void sum (int first, int second) { return first + second; }
```

題解

選項 A，少了將結果回傳的「return」。

選項 B，方法的第二個參數少了型態。

選項 C，少了方法回傳值的型態。

選項 D，正確答案。

選項 E，因為要有回傳值，所以不能使用「void」。

(40)

Person.java
```
public class Person {
    String name;
    int age;

    public Person(String n, int a) {
        name = n;
        age = a;
    }

    public String getName() {
        return name;
    }
```

```
    public int getAge() {
        return age;
    }
}
```

Test.java

```
public class Test {
    public static void checkAge(List<Person> list,
Predicate<Person> predicate) {
        for (Person p : list) {
            if (predicate.test(p)) {
                System.out.println(p.name + " ");
            }
        }
    }

    public static void main(String[] args) {
        List<Person> iList = Arrays.asList(new Person("Hank",
45), new Person("Charlie", 40), new Person("Smith", 38));
        // line n1
    }
}
```

Which code fragment, when inserted at line n1, enables the code to print Hank?

A
```
checkAge(iList, () -> p.getAge() > 40);
```

B
```
checkAge(iList, Person p -> p.getAge() > 40);
```

C(ans)
```
checkAge(iList, p -> p.getAge() > 40);
```

D
```
checkAge(iList, (Person p) -> { p.getAge() > 40; });
```

題解

Predicate 的 JDK 原始碼片段如下：

```
@FunctionalInterface
public interface Predicate<T> {

    boolean test(T t);
}
```

要實體化 Predicate 類別，必須要實作 test 方法。選項 C 是正確的 Lambda 語法的用法，可以實作出 Predicate 類別的 test 方法。

(41)

```
for (int ii = 0; ii < 3; ii++) {
    int count = 0;
    for (int jj = 3; jj > 0; jj--) {
        if (ii == jj) {
            ++count;
            break;
        }
```

```
        }
        System.out.print(count);
        continue;
    }
```

What is the result?

A(ans)
011

B
012

C
123

D
000

題解
外層 for 迴圈會執行三次。
第一次執行外層 for 迴圈，ii=0。內層 for 迴圈，jj 的計數範圍是 3~1，共執行 3 次。在這次外迴圈的執行中，jj 都不會等於 ii，因此第一次外層迴圈執行到最後，count 變數的值不會被改變，會輸出「0」。
第二次執行外層 for 迴圈，ii=1。內層 for 迴圈，jj 的計數範圍是 3~1，執行到第三次時，jj 會等於 ii，所以要將 count 加 1，然後立刻跳出內層 for 迴圈。第二次外層迴圈執行到最後，會輸出「1」。
第三次執行外層 for 迴圈，ii=2。內層 for 迴圈，jj 的計數範圍是 3~1，執行到第二次時，jj 會等於 ii，所以要將 count 加 1，然後立刻跳出內層 for 迴圈。第三次外層迴圈執行到最後，會輸出「1」。

(42)
Given:

```
public class MainTest {
    public static void main(int[] args) {
        System.out.println("int main " + args[0]);
    }

    public static void main(Object[] args) {
        System.out.println("Object main " + args[0]);
    }

    public static void main(String[] args) {
        System.out.println("String main " + args[0]);
    }
}
```

and commands:

```
javac MainTest.java
java MainTest 1 2 3
```

What is the result?

A
int main 1

B
Object main 1

C(ans)
String main 1

D
Compilation fails

E
An exception is thrown at runtime
題解
程式進入點為「public static void main(String[] args)」或是「public static void main(String... args)」，因此這題會執行第 11 行的 main 方法。

(43)
Which statement best describes encapsulation?
A(ans)
Encapsulation ensures that classes can be designed so that only certain fields and methods of an object are accessible from other objects.

B
Encapsulation ensures that classes can be designed so that their methods are inheritable.

C
Encapsulation ensures that classes can be designed with some fields and methods declared as abstract.

D
Encapsulation ensures that classes can be designed so that if a method has an argument MyType x, any subclass of MyType can be passed to that method.
題解
選項 A，封裝可以確保類別物件中，只有特定的欄位和方法能被其他物件存取。
選項 B，封裝的方法不一定需要被繼承，封裝和繼承沒有什麼直接的關聯性。
選項 C，欄位不能是抽象的。
選項 D，這個敘述應該是多型的概念。

(44)
```
String[] cartoons = {"tom", "jerry", "micky", "tom"};
int counter = 0;
if ("tom".equals(cartoons[0])) {
    counter++;
} else if ("tom".equals(cartoons[1])) {
    counter++;
} else if ("tom".equals(cartoons[2])) {
    counter++;
} else if ("tom".equals(cartoons[3])) {
    counter++;
}
System.out.print(counter);
```

What is the result?

A(ans)
1

B
2

C
4

D
0

題解
cartoons[0]所參考到的字串物件是「tom」，所以第一個 if 的條件式會成立，將 counter 變數的值加 1。由於之後其它條件判斷都是使用 else if，因此當前面有一個條件式成立之後，就不會再繼續判斷下去了，所以最後輸出「1」。

(45)
```
public class Student {
    public String name = "";
    public int age = 0;
    public String major = "Undeclared";
    public boolean fulltime = true;

    public void display() {
        System.out.println("Name: " + name + " Major: " +
major);
    }

    public boolean isFullTime() {
        return fulltime;
    }
}
```

Which line of code initializes a student instance?

A
Student student1;

B
Student student1 = Student.new();

C(ans)
Student student1 = new Student();

D
Student student1 = Student();

題解
選項 A，僅僅只是宣告出一個可以用來儲存 String 物件參考的 student1 變數。
選項 B，如果是要將「new」當作方法使用，應該使用物件類別的「newInstance」方法，用法如下：
```
try {
    Student student1 = Student.class.newInstance();
```

```
} catch (InstantiationException ex) {

} catch (IllegalAccessException ex) {

}
```
選項 C，正確實體化物件的方式。
選項 D，少了「new」運算子。

(46)
```
abstract class Planet {
    protected void revolve() { // line n1

    }
    abstract void rotate(); // line n2
}

class Earth extends Planet {
    void revolve() { // line n3

    }

    protected void rotate() { // line n4
    }
}
```

Which two modifications, made independently, enable the code to compile?

A
Make the method at line n1 public.

B
Make the method at line n2 public.

C(ans)
Make the method at line n3 public.

D(ans)
Make the method at line n3 protected.

E
Make the method at line n4 public.

題解
題目原本提供的程式會在 line n3 編譯錯誤，原因在於 Earth 類別繼承的 Planet 類別，其 revolve 方法的可見度使用 protected 來修飾。若要在 Earth 類別覆寫 revolve 方法，其可見度必須不能小於 protected，因此 Earth 類別的 revolve 方法的可見度只能用 protected 或是 public 來修飾。

(47)
```
public static void main(String[] args) {
    String str = " ";
    str.trim();
    System.out.println(str.equals("") + " " + str.isEmpty());
}
```
What is the result?

A
true true

B
true false

C(ans)
false false

D
false true
題解
String 物件的 trim 方法可以消除開頭和結尾的所有空格，回傳新的字串，並不會改變原有字串物件的內容。所以在這題中，第 7 行執行完後，str 所參考到的字串物件依然是「 」。

(48)
```
public class Painting {
    private String type;

    public String getType() {
        return type;
    }

    public void setType(String type) {
        this.type = type;
    }

    public static void main(String[] args) {
        Painting obj1 = new Painting();
        Painting obj2 = new Painting();
        obj1.setType(null);
        obj2.setType("Fresco");
        System.out.print(obj1.getType() + " : " +
obj2.getType());
    }
}
```
What is the result?
A
: Fresco

B(ans)
null : Fresco

C
Fresco : Fresco

D
A NullPointerException is thrown at runtime
題解
使用「+」運算子時，若其中至少一個運算元為字串(物件)型態，則為字串連接運算。若連接的字串(物件)參考是 null 的話，則會被當作長度為 4 的「null」字串。

(49)

```
class Dog {
    Dog() {
        try {
            throw new Exception();
        } catch (Exception e) { }
    }
}

class Test {
    public static void main(String[] args) {
        Dog d1 = new Dog();
        Dog d2 = new Dog();
        Dog d3 = d2;
        // do complex stuff
    }
}
```
How many objects have been created when the line // do complex stuff is reached?

A
Two

B
Three

C(ans)
Four

D
Six

題解

這題目是在問當執行到「// do complex stuff」這行的時候，已經建立了多少的物件了。注意這裡指的物件不是只有 Dog 物件，所有的物件都要計入。

程式第 13 行和第 14 行，都建立出了一個 Dog 物件。建立 Dog 物件時，會執行第 3 行的建構子，並在 try-catch 內建立並拋出新的 Exception 物件。因此每建立一次 Dog 物件，實際上會建立出兩個物件，建立兩次 Dog 物件，就會建立四個物件。

(50)
```
public class App {
    public static void main(String[] args) {
        Boolean[] bool = new Boolean[2];
        bool[0] = new Boolean(Boolean.parseBoolean("true"));
        bool[1] = new Boolean(null);
        System.out.println(bool[0] + " " + bool[1]);
    }
}
```
What is the result?

A(ans)
true false

B
true null

C
Compilation fails

D
A NullPointerException is thrown at runtime
題解
程式第 6 行，會產生內容為「true」的 Boolean 物件。
程式第 7 行，在 Boolean 類別的建構子參數傳入 null 會產生內容為「false」的 Boolean 物件，這部份可以參考 Java 的原始碼：

```
public Boolean(boolean value) {
    this.value = value;
}

public Boolean(String s) {
    this(parseBoolean(s))
}
public static boolean parseBoolean(String s) {
    return ((s != null) && s.equalsIgnoreCase("true"));
}
```

(51)

```
public class Person {

    String name;
    int age = 25;

    public Person(String name) {
        this(); // line n1
        setName(name);
    }

    public Person(String name, int age) {
        Person(name); // line n2
        setAge(age);
    }

    //setter and getter methods go here

    public String show() {
        return name + " " + age;
    }

    public static void main(String[] args) {
        Person p1 = new Person("Jesse");
        Person p2 = new Person("Walter", 52);
        System.out.println(p1.show());
        System.out.println(p2.show());
    }
}
```

What is the result?

A
Jesse 25
Walter 52

B
Compilation fails only at line n1

C
Compilation fails only at line n2

D(ans)
Compilation fails at both line n1 and line n2
題解
這題的 line n1 會發生編譯錯誤，因為 Person 類別並沒有無參數的建構子。line n2 也會編譯錯誤，因為 Person(String name)是建構子，沒有辦法直接這樣呼叫，應改成「this(name)」。

(52)
DoInterface.java
```
package p1;

public interface DoInterface {

    void method1(int n1); // line n1
}
```

Test.java
```
package p3;

import p1.DoInterface;

class DoClass implements DoInterface {

    public DoClass(int p1) {
    }

    public void method1(int p1) {} // line n2

    private void method2(int p1) {} // line n3
}

public class Test {

    public static void main(String[] args) {
        DoClass doi = new DoClass(100); // line n4
        doi.method1(100);
        doi.method2(100);
    }
}
```

Which change will enable the code to compile?

A
Adding the public modifier to the declaration of method1 at line n1

B
Removing the public modifier from the definition of method1 at line n2

C(ans)

Changing the private modifier on the declaration of method 2 public at line n3

D
Changing the line n4 DoClass doi = new DoClass();
題解
題目原先提供的程式會編譯錯誤，因為 DoClass 類別的 method2 方法使用了 private 修飾字來修飾，是私有成員，只有在 DoClass 類別內才可以存取。因此 Test 類別的 main 方法若要存取 DoClass 類別物件的 method2 方法會發生編譯錯誤。
選項 A，介面(interface)定義的方法會自動以 public 修飾字來修飾，因此在 line n1 加上 public 修飾的結果和沒加 public 修飾的結果是一樣的。
選項 B，移除 line n2 的 public 修飾字並不能改變 method2 只能在 DoClass 類別使用的情形。
選項 C，使用 public 來修飾 method2 方法，讓它可以在 DoClass 類別之外被使用，為正確答案。
選項 D，DoClass 類別並沒有無參數的建構子，這樣修改反而會造成更多的編譯問題。

(53)
Which two statements correctly describe checked exception?

A(ans)
These are exceptional conditions that a well-written application should anticipate and recover from.

B
These are exceptional conditions that are external to the application, and that the application usually cannot anticipate or recover from.

C
These are exceptional conditions that are internal to the application, and that the application usually cannot anticipate or recover from.

D
Every class that is a subclass of RuntimeException and Error is categorized as checked exception.

E(ans)
Every class that is a subclass of Exception, excluding RuntimeException and its subclasses, is categorized as checked exception.
題解
選項 A，checked exception 在撰寫程式的時候一定要被處理，所以這個敘述是正確的。
選項 B，這是 Error 的敘述。
選項 C，這是 RuntimeException 的敘述。RuntimeException 為 unchecked exception。
選項 D，RuntimeException 和 Error 以及繼承它們的例外和錯誤都是 unchecked exception 才對。
選項 E，除了 RuntimeException 和繼承 RuntimeException 的例外之外，Exception 和其它繼承 Exception 的例外都是 checked exception。

(54)

```
public class TestLoop {

    public static void main(String[] args) {
        int array[] = {0, 1, 2, 3, 4};
        int key = 3;
        for (int pos = 0; pos < array.length; ++pos) {
            if (array[pos] == key) {
                break;
            }
        }
        System.out.print("Found " + key + " at " + pos);
    }
}
```

What is the result?

A
Found 3 at 2

B
Found 3 at 3

C(ans)
Compilation fails

D
An exception is thrown at runtime
題解
這題的程式會編譯錯誤，因為 pos 變數是在 for 迴圈內被宣告的，但是程式第 11 行卻要去取得 pos 變數的值，因此編譯時會有無法找到變數的錯誤。

(55)

```
public static void main(String[] args) {
    StringBuilder sb = new StringBuilder(5);
    String s = "";

    if (sb.equals(s)) {
        System.out.println("Match 1");
    } else if (sb.toString().equals(s.toString())) {
        System.out.println("Match 2");
    } else {
        System.out.println("No Match");
    }
}
```

What is the result?

A
Match 1

B(ans)
Match 2

C
No Match

D
A NullPointerException is thrown at runtime.
題解
程式第 7 行，判斷 StringBuilder 物件和字串物件邏輯上是否相等。它們是不同類型的物件，永遠不會相等。
程式第 8 行，判斷 StringBuilder 物件轉成字串後和字串物件邏輯上是否相等。它們都是空字串，因此相等，輸出「Match 2」。

(56)
```
int[] intArr = {8, 16, 32, 64, 128};
```

Which two code fragments, independently, print each element in this array?

A
```
for (int i : intArr) {
    System.out.print(intArr[i] + " ");
}
```

B(ans)
```
for (int i : intArr) {
    System.out.print(i + " ");
}
```

C
```
for (int i = 0; intArr) {
    System.out.print(intArr[i] + " ");
    i++;
}
```

D
```
for (int i = 0; i < intArr.length; i++) {
    System.out.print(i + " ");
}
```

E(ans)
```
for (int i = 0; i < intArr.length; i++) {
    System.out.print(intArr[i] + " ");
}
```
題解
選項 A，在第一次執行 for 迴圈時會拋出 ArrayIndexOutOfBoundsException，因為 intArr[8]超出陣列索引範圍。
選項 B，正確的 foreach 用法。
選項 C，在第一次執行 for 迴圈時會拋出 ArrayIndexOutOfBoundsException，因為 intArr[8]超出陣列索引範圍。
選項 D，會輸出「0 1 2 3 4 」，不是我們要的結果。
選項 E，正確的程式。

(57)
Which of the following can fill in the blank in this code to make it compile? (Select 2 options.)

```
public void method() ____ Exception{
```

```
    _____ Exception();
}
```

A(ans)
On line 1, fill in throws

B
On line 1, fill in throws new

C(ans)
On line 2, fill in throw new

D
On line 2, fill in throws

E
On line 2, fill in throws new

題解
要讓方法將例外往外拋出，需使用「throws」關鍵字。要拋出一個新的例外，需使用「throw」關鍵字。要實體化出一個例外，需使用「new」運算子

(58)
```
interface Contract {
}

class Super implements Contract {
}

class Sub extends Super {
}

public class Ref {

    public static void main(String[] args) {
        List objs = new ArrayList();
        Contract c1 = new Super();
        Contract c2 = new Sub(); // line n1
        Super s1 = new Sub();

        objs.add(c1);
        objs.add(c2);
        objs.add(s1); // line n2

        for (Object itm : objs) {
            System.out.println(itm.getClass().getName());
        }
    }
}
```

What is the result?

A(ans)
Super
Sub
Sub

B
Contract
Contract
Super

C
Compilation fails at line n1

D
Compilation fails at line n2
題解
這題的 ArrayList 物件並沒有使用到泛型，因此 ArrayList 物件可以存入任意的 Object 物件。
物件的 getClass 方法會回傳物件實體所屬的類別。

(59)
```
int nums1[]  = new int[3];
int nums2[]  = {1,2,3,4,5};
nums1 = nums2;
for(int x : nums1){
    System.out.print(x + ":");
}
```

What is the result?

A(ans)
1:2:3:4:5:

B
1:2:3:

C
Compilation fails.

D
An ArrayoutofBoundsException is thrown at runtime.
題解
程式第 7 行，會將 nums2 的陣列物件參考指派給 nums1 儲存，因此 nums1 和 nums2 變數所存的物件參考都是一樣的，表示為同一個物件。
程式第 8 行，用 foreach 的方式取得 nums1 陣列物件的所有元素，並將它們全部印出。
(60)
View the exhibit.
```
class MissingInfoException extends Exception {
}

class AgeOutofRangeException extends Exception {
}

class Candidate {

    String name;
    int age;

    Candidate(String name, int age) throws Exception {
        if (name == null) {
```

```
            throw new MissingInfoException();
        } else if (age <= 10 || age >= 150) {
            throw new AgeOutofRangeException();
        } else {
            this.name = name;
            this.age = age;
        }
    }

    public String toString() {
        return name + " age: " + age;
    }
}
```

Given the code fragment:

```
public class Test {
    public static void main(String[] args) {
        Candidate c = new Candidate("James", 20);
        Candidate c1 = new Candidate("Williams", 32);
        System.out.println(c);
        System.out.println(c1);
    }
}
```

Which change enables the code to print the following?

```
James age: 20
Williams age: 32
```

A
Replacing line 5 with
```
public static void main (String [] args) throws MissingInfoExc
eption, AgeOutofRangeException {
```

B
Replacing line 5 with
```
public static void main (String [] args) throws.Exception {
```

C(ans)
Enclosing line 6 and line 7 within a try block and adding:
```
catch (MissingInfoException e1) { //code goes here
}
catch (AgeOutofRangeException e2) {//code goes here
}
catch(Exception e3) { //code goes here
}
```

D
Enclosing line 6 and line 7 within a try block and adding:
```
catch (MissingInfoException e2) { //code goes here
}
catch (AgeOutofRangeException e3) {//code goes here
}
```

題解
原本題目提供的程式會在第 6 行和第 7 行出現編譯錯誤，原因在於 Candidate 的物件有拋出 Exception。由於 Exception 是需要檢查的例外(checked exception)，因此必須要撰寫程式去處理它。
選項 A，只讓 main 方法拋出 MissingInfoException 和 AgeOutofRangeException 是不夠的，必須要拋出 Exception 才行。
選項 B，throws 的用法錯誤。
選項 C，有 catch 到 Exception，正確選項。
選項 D，只讓 try-catch 接住 MissingInfoException 和 AgeOutofRangeException 是不夠的，必須要接住 Exception 才行。

(59)

```
public class Case{
    public static void main(String[] args){
        String product = "Pen";
        product.toLowerCase();
        product.concat(" BOX".toLowerCase());
        System.out.print(product.substring(4, 6));
    }
}
```

What is the result?

A
box

B
nbo

C
bo

D
nb

E(ans)

An exception is thrown at runtime

題解
字串物件的 toLowerCase、concat 和 substring 方法都不會影響到字串物件本身的內容，而是會回傳出新的字串物件。
substring 方法可以取得字串中某一段子字串，在這題中要取 product 字串變數所參考到的「Pen」字串中，索引 4 到索引 6(不包含索引 6)的子字串，由於「Pen」字串的長度沒那麼長，因此會拋出 StringIndexOutOfBoundsException 例外。

(60)

```
List<String> names = new ArrayList<>();
names.add("Robb");
names.add("Bran");
names.add("Rick");
names.add("Bran");
if (names.remove("Bran")) {
    names.remove("Jon");
}
```

```
System.out.println(names);
```

What is the result?

A(ans)
[Robb, Rick, Bran]

B
[Robb, Rick]

C
[Robb, Bran, Rick, Bran]

D
An exception is thrown at runtime.

題解
List 集合允許擁有多個邏輯相同的元素，remove 方法會從頭尋找第一個符合的元素，並將其移除。若傳入 remove 方法的參數為要移除的物件，則會回傳該物件是否有在集合裡面；若傳入 remove 方法的參數為索引值，則會回傳索引值對應被移除的物件。
在這題中，要移除一個「Bran」字串物件和一個「Jon」字串物件，但「Jon」字串物件本來就不存在於集合中，因此實際上只會移除一個「Bran」字串物件。

(61)
```
int var1 = -5;
int var2 = var1--;
int var3 = 0;
if (var2 < 0) {
    var3 = var2++;
} else {
    var3 = --var2;
}
System.out.println(var3);
```

What is the result?

A
-6

B
-4

C(ans)
-5

D
5

E
4

F
Compilation fails

題解
程式第 7 行，將 var1 的值指派給 var2 變數後，再把 var1 變數的值減 1。此時 var1=-6，var2=-5。

程式第 9 行 if 條件式成立，因此會執行第 10 行程式，將 var2 的值指派給 var3 變數後，再把 var2 變數的值加 1。此時 var3=-5，var2=-4。
程式第 14 行，輸出 var3 的值「-5」。

(62)
Which statement is/are true?
I. Default constructor only contains "super();" call.
II. We can't use any access modifier with a constructor.
III. A constructor should not have a return type.

A
Only I.

B
Only II.

C
Only I and II.

D(ans)
Only I and III.

E
ALL
題解
敘述 I 是正確的。
敘述 II 是錯誤的，建構子也可以使用 public、pretected、private 等等的修飾字來進行可見度的調整。
敘述 III 是正確的，建構子不能定義回傳值的型態。

(63)
```
public static void main(String[] args) {
    int num = 5;
    do {
        System.out.print(num-- + " ");
    } while (num == 0);
}
```

What is the result?

A
5 4 3 2 1 0

B
5 4 3 2 1

C
4 2 1

D(ans)
5

E
Nothing is printed

題解
do-while 迴圈第一次執行，輸出「5 」並將 num 變數所儲存的數值再減一，num=5-1=4。4 不等於 0，因此 while 條件式不成立，迴圈不再執行。所以只有輸出「5 」。

(64)
```
public static void main(String[] args) {
    String[] planets = {"Mercury", "Venus", "Earth", "Mars"};

    System.out.println(planets.length);
    System.out.println(planets[1].length());
}
```

What is the output?

A
4
4

B
3
5

C
4
7

D
5
4

E(ans)
4
5

F
4
21

題解
planets 是一個長度為 4 的字串陣列。planets[1]所參考到的字串物件是「Venus」，長度為 5。

(65)
Which code fragment cause a compilation error?

A
```
float flt = 100F;
```

B
```
float flt = (float) 1_11.00;
```

C
```
float flt = 100;
```

D(ans)
```
double y1 = 203.22;
```

float flt = y1;

E
int y2 = 100;
float flt = (float) y2;
題解
選項 D 會編譯錯誤，因為 double 不能隱含式(implicit)轉成 float。

(66)
Integer number = Integer.valueOf("808.1");

Which is true about the above statement?

A
The value of the variable number will be 808.1

B
The value of the variable number will be 808

C
The value of the variable number will be 0.

D(ans)
A NumberFor**matE**x**c**eption will be throw.

E
It will not compile.
題解
Integer 為整數的 Wrapper 類別，若是想把不是整數的字串轉成 Integer 物件，會拋出 NumberFormatException 例外。

(67)
Which statement will empty the contents of a StringBuilder variable named sb?

A
sb.deleteAll();

B
sb.delete(0, sb.size());

C(ans)
sb.delete(0, sb.length());

D
sb.removeAll();
題解
選項 A，StringBuilder 物件沒有 deleteAll 方法。
選項 B，StringBuilder 物件的 delete 方法可以刪除某範圍內的字元。但是 StringBuilder 物件並沒有 size 方法，如果是要取得字串長度，應用 length 方法。
選項 C，正確答案。
選項 D，StringBuilder 物件沒有 removeAll 方法。

(68)

Which of the following exception will be thrown due to the statement given here?

```
int array[] = new int[-2];
```

A
NullPointerException

B(ans)
NegativeArraySizeException

C
ArrayIndexOutOfBoundsException

D
IndexOutOfBoundsException

E
This statement does not cause any exception.
題解
由於在實體化陣列時定義的陣列長度小於 0，因此會拋出 NegativeArraySizeException 例外。

(69)
Acc.java

```
package p1;

public class Acc {

    int p;
    private int q;
    protected int r;
    public int s;
}
```

Test.java

```
package p2;

import p1.Acc;

public class Test extends Acc {

    public static void main(String[] args) {
        Acc obj = new Test();
    }
}
```

Which statement is true?

A
Both p and s are accessible by obj.

B(ans)
Only s is accessible by obj.

C

Both r and s are accessible by obj.

D
p, r, and s are accessible by obj.
題解
在不同套件的繼承關係下，只有 public 和 protected 的可見度可以將變數繼承至底下的類別。因此在這題中，只有 s 變數和 r 變數可以用 Test 類別所實體化出來的 Test 物件存取。但是，obj 變數是 Acc 型態，不是 Test 型態，因此沒有辦法存取到使用 protected 修飾的 r 變數。

(70)
```
public class TestLoop1 {

    public static void main(String[] args) {
        int a = 0, z = 10;
        while (a < z) {
            a++;
            --z;
        }
        System.out.print(a + " : " + z);
    }
}
```

What is the result?

A(ans)
5:5

B
6:4

C
6:5

D
5:4
題解
while 迴圈第一次執行，0 小於 10，條件式成立，a=0+1=1，z=10-1=9。
第二次執行，1 小於 9，條件式成立，a=1+1=2，z=9-1=8。
第三次執行，2 小於 8，條件式成立，a=2+1=3，z=8-1=7。
第四次執行，3 小於 7，條件式成立，a=3+1=4，z=7-1=6。
第五次執行，4 小於 6，條件式成立，a=4+1=5，z=6-1=5。
第六次執行時，5 不小於 5，所以跳出 while 迴圈，輸出「5 : 5」。

(71)
```
boolean log3 = (5.0 != 6.0) && (4 != 5);
boolean log4 = (4 != 4) || (4 == 4);
System.out.println("log3:" + log3 + "\nlog4" + log4);
```

What is the result?

A

```
log3:false
log4:true
```

B(ans)
log3:true
log4:true

C
```
log3:true
log4:false
```

D
```
log3:false
log4:false
```

題解
「!=」是關係運算子，表示不等於。
「&&」是邏輯運算子，表示 AND，當兩個運算元都是 true 的時候回傳 true，在計算的時候若第一個運算元為 false，則不再繼續執行第二個運算元，直接回傳 false。
「||」是邏輯運算子，表示 OR，當兩個運算元都是 false 的時候回傳 false，在計算的時候若第一個運算元為 true，則不再繼續執行第二個運算元，直接回傳 true。
```
log3 = (5.0 != 6.0) && (4 != 5) = true && (4 != 5) = true &&
true = true

log4 = (4 != 4) || (4 == 4) = false || (4 == 4) = false ||
true = true
```

(72)
```
System.out.println(2 + 4 * 9 - 3); //Line 21
System.out.println((2 + 4) * 9 - 3); // Line 22
System.out.println(2 + (4 * 9) - 3); // Line 23
System.out.println(2 + 4 * (9 - 3)); // Line 24
System.out.println((2 + 4 * 9) - 3); // Line 25
```

Which line of codes prints the highest number?

A
Line 21

B(ans)
Line 22

C
Line 23

D
Line 24

E
Line 25

題解
Java 程式的算式遵循「先乘除後加減」與「括號內先計算」的四則運算規擇。
第 21 行的算法為：
```
2 + 4 * 9 - 3 = 2 + 36 - 3 = 38 - 3 = 35
```
第 22 行的算法為：
```
(2 + 4) * 9 - 3 = 6 * 9 - 3 = 54 - 3 = 51
```

第 23 行的算法為：
2 + (4 * 9) - 3 = 2 + 36 - 3 = 38 - 3 = 35
第 24 行的算法為：
2 + 4 * (9 - 3) = 2 + 4 * 6 = 2 + 24 = 26
第 25 行的算法為：
(2 + 4 * 9) - 3 = (2 + 36) - 3 = 38 - 3 = 35

(73)

```
interface Runnable{
    public void run();
}
```

Which of the following will create instance of Runnable type?

A(ans)

```
Runnable run = () -> {System.out.println("Run");};
```

B

```
Runnable run = () -> System.out.println("Run");
```

C

```
Runnable run = () > System.out.println("Run");
```

D

```
Runnable run = > System.out.println("Run");
```

E
None of the above.
題解
Java 8 導入了 Lambda 語法

(73)

MyException.java

```
public class MyException extends RuntimeException{

}
```

Test.java

```
public class Test {

    public static void main(String[] args) {
        try {
            method1();
        } catch (MyException ne) {
            System.out.print("A");
        }
    }

    public static void method1() { // line n1
        try {
            throw Math.random() > 0.5 ? new MyException() :
new RuntimeException();
        } catch (RuntimeException re) {
            System.out.print("B");
        }
    }
```

```
}
```

What is the result?

A
A

B(ans)
B

C
Either A or B

D
A B

E
A compile time error occurs at line n1

題解
MyException 是 RuntimeException 的一種，因此第 13 行無論拋出 MyException 還是 RuntimeException，都會在第 14 行被 catch 接住，而輸出「B」。

(74)

```
public class SampleClass {

    public static void main(String[] args) {
        AnotherSampleClass asc = new AnotherSampleClass();
        SampleClass sc = new SampleClass();
        sc = asc;
        System.out.println("sc: " + sc.getClass());
        System.out.println("asc: " + asc.getClass());
    }
}

class AnotherSampleClass extends SampleClass {
}
```

What is the result?

A
sc: class Object
asc: class AnotherSampleClass

B
sc: class SampleClass
asc: class AnotherSampleClass

C
sc: class AnotherSampleClass
asc: class SampleClass

D(ans)
sc: class AnotherSampleClass
asc: class AnotherSampleClass

題解
程式第 5 行實體化出來的 SampleClass 物件，會在第 6 行執行完後不被任何的變數參考到，會在稍候被垃圾回收(Garbage Collection)。
執行程式第 7 行時，sc 和 asc 都是同一個 AnotherSampleClass 物件實體，因此呼叫 getClass 方法會得到相同的結果。

(75)
Given:
```
class Jump {

    static String args[] = {"lazy", "lion", "is", "always"};

    public static void main(String[] args) {
        System.out.println(args[1] + " " + args[2] + " " +
args[3] + " jumping");
    }
}
```

And the commands:
```
javac Jump.java
java Jump crazy elephant is always
```

What is the result?

A
lazy lion is jumping

B
lion is always jumping

C
crazy elephant is jumping

D(ans)
elephant is always jumping

E
Compilation fails

題解
題目執行的指令會編譯「Jump.java」，由於 Jump 類別有用 public 修飾，因此預設會執行 Jump 類別的 main 方法，並把「crazy elephant is always」作為 main 的參數。參數的索引值從 0 開始。
main 方法的 args 參數會遮蔽(shadow)掉 Jump 類別的 args 類別(靜態)變數。
因此會輸出「elephant is always jumping」。

(76)
```
package p1;

public class Test {

    static double dvalue;
    static Test ref;

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

```
        System.out.println(ref);
        System.out.println(dvalue);
    }
}
```

What is the result?

A
p1.Test.class
0.0

B
0.000000

C(ans)
null
0.0

D
Compilation fails

E
A NullPointerException is thrown at runtime

題解
dvalue 和 ref 是 Test 的類別變數，在宣告時就會自動被初始化。dvalue 的型態為 double，會被初始化為 0。ref 的型態為 Test，為物件參考型態，會被初始化為 null。println 方法若傳入的參考為 null，會直接輸出「null」。
因此這題答案是選項 C。

(77)
DoInterface.java
```
package p1;

public interface DoInterface {

    void m1(int n); // line n1

    public void m2(int n);
}
```

DoClass.java
```
package p3;
public class DoClass implements DoInterface{
    int x1, x2;
    DoClass(){
        this.x1 = 0;
        this.x2 = 10;
    }
    public void m1(int p1) { x1+=p1; System.out.println(x1); }
// line n2
    public void m2(int p1) { x2+=p1; System.out.println(x2); }
}
```

Test.java
```
package p2;
```

```
import p1.*;
import p3.*;

class Test {

    public static void main(String[] args) {
        DoInterface doi = new DoClass(); // line n3
        doi.m1(100);
        doi.m2(200);
    }
}
```

What is the result?

A
100
210

B
Compilation fails due to an error in line n1

C
Compilation fails due to an error at line n2

D(ans)
Compilation fails due to an error at line n3
題解
line n3 實體化 DoClass 類別的物件實體，但是由於 DoClass 的建構子並沒有使用 public 修飾字宣告，因此它只能在同一個套件(package)之內被實體化。若在不同的套件下實體化 DoClass 物件，會發生編譯錯誤。

(78)

```
public class App {

    public static void main(String[] args) {
        String str1 = "Java";
        String str2 = new String("java");
        //line n1
        {
            System.out.println("Equal");
        } else {
            System.out.println("Not Equal");
        }
    }
}
```

Which code fragment, when inserted at line n1, enables the App class to print Equal?

A
```
String str3 = str2;
if (str1 == str3)
```

B(ans)
if (str1.equalsIgnoreCase(str2))

C

```
String str3 = str2;
if (str1.equals(str3))
```

D
```
if (str1.toLowerCase() == str2.toLowerCase())
```
題解
Java 的「==」運算子會判斷運算元的值是否相等，而非內容是否相等。在這裡若要輸出「Equal」，必須要讓 if 條件式成立。
選項 A，str2 和 str3 都是參考至相同的物件，但 str1 和 str2 是兩個不同的物件，所以它們的參考值是不一樣的，if 條件式不會成立。
選項 B，equalsIgnoreCase 方法可以判斷字串在忽略字母大小寫的情況下是否相等。str1 是「Java」，str2 是「java」，因此 if 條件式成立。
選項 C，str2 和 str3 都是參考至相同的物件，equals 可以判斷字串是否完全相等。str1 是「Java」，str3 是「java」，因此 if 條件式不成立。
選項 D，toLowerCase 方法會回傳將字串內的字母全都轉成小寫字母的結果。str1.toLowerCase() 會回傳「java」，str2.toLowerCase() 會回傳「java」，但雖然這兩字串都是「java」，它們實際上是不同的物件，因此有不同的參考，所以經過「==」運算後，結果將為 false。

(79)
```
class StaticField {

    static int i = 7;

    public static void main(String[] args) {
        StaticField obj = new StaticField();
        obj.i++;
        StaticField.i++;
        obj.i++;
        System.out.println(StaticField.i + " " + obj.i);
    }
}
```

What is the result?

A(ans)
10 10

B
8 9

C
9 8

D
7 10
題解
i 為類別(靜態)變數，因此不需考慮到物件的實體，因為不管實體是什麼，存取到的 i 變數都是類別的 i 變數。
一開始 i 變數的值為 7，後來加了 3 次 1，因此最後 i 變數的值為 10。

(80)
Which three are advantages of the Java exception mechanism?

A(ans)
Improves the program structure because the error handling code is separated from the normal program function

B
Provides a set of standard exceptions that covers all the possible errors

C(ans)
Improves the program structure because the programmer can choose where to handle exceptions

D
Improves the program structure because exceptions must be handled in the method in which they occurred

E(ans)
Allows the creation of new exceptions that are tailored to the particular program being created

題解
選項 A，分割一般程式和例外處理程式，能讓程式的結構更淺顯易懂。
選項 B，Java 內建多種例外種類，只包含程式實作層面可能會發生的錯誤，並非全部的錯誤，如因硬體損壞而出現難以辨識的錯誤或是程式邏輯上的錯誤。
選項 C，Java 的程式設計師可以自行決定程式的哪些部份要進行例外處理。
選項 D，例外不一定要在方法內進行處理，也可以將其丟出方法之外，交由執行緒來進行中斷。
選項 E，可以針對特定程式建立新的例外。

(81)
```
public static void main(String[] args) {
    String date = LocalDate.parse("2014-05-
04").format(DateTimeFormatter.ISO_DATE_TIME);
    System.out.println(date);
}
```

What is the result?

A
May 04, 2014T00:00:00.000

B
2014-05-04T00:00:00.000

C
5/4/14T00:00:00.000

D(ans)
An exception is thrown at runtime.

題解
程式第 7 行會使用 parse 方法產生 LocalDate 物件，再用 LocalDate 物件的 format 方法將 LocalDate 物件轉成字串，但此時會拋出 UnsupportedTemporalTypeException 例外。原因在於 ISO_DATE_TIME 這個 DateTimeFormatter 無法用於 LocalDate 物件，LocalDate 物件並沒有時間的資訊。

若改成使用 ISO_DATE 這個 DateTimeFormatter，則輸出結果為：
2014-05-04

(81)
A method is declared to take three arguments. A program calls this method and passes only two arguments. What is the results?

A(ans)
C**ompilation fails.**

B
The third argument is given the value null.

C
The third argument is given the value void.

D
The third argument is given the value zero.

E
The third argument is given the appropriate falsy value for its declared type. F) An exception occurs when the method attempts to access the third argument.
題解
呼叫方法若無法從已定義的方法中找到符合的簽名(signature)，會發生編譯錯誤。

(82)
Given the following code for a Planet object:

```
public class Planet {

    public String name;
    public int moons;

    public Planet(String name, int moons) {
        this.name = name;
        this.moons = moons;
    }
}
```

And the following main method:

```
public static void main(String[] args) {
    Planet[] planets = {
        new Planet("Mercury", 0),
        new Planet("Venus", 0),
        new Planet("Earth", 1),
        new Planet("Mars", 2)
    };

    System.out.println(planets);
    System.out.println(planets[2]);
    System.out.println(planets[2].moons);
}
```

What is the output?

A

```
planets
Earth
1

B
[LPlanets.Planet;@15db9742
Earth
1

C(ans)
[LPlanets.Planet;@15db9742
Planets.Planet@6d06d69c
1

D
[LPlanets.Planet;@15db9742
Planets.Planet@6d06d69c
[LPlanets.Moon;@7852e922

E
[LPlanets.Planet;@15db9742
Venus
0
```

題解

println 方法會輸出傳入參數的 toString 方法回傳的字串。
程式第 9 行，要印出 Planet 陣列，所以字串會是「[L」加上類別的路徑和雜湊值。
程式第 10 行，要印出 Planet 物件，所以字串會類別的路徑和雜湊值。
程式第 11 行，要印出 Planet 物件的 moons 這個整數型態的欄位，所以會輸出整數。
所以答案是選項 C。

(83)

Test.java

```
class Alpha {

    public String[] main = new String[2];

    Alpha(String[] main) {
        for (int ii = 0; ii < main.length; ii++) {
            this.main[ii] = main[ii] + 5;
        }
    }

    public void main() {
        System.out.print(main[0] + main[1]);
    }
}

public class Test {

    public static void main(String[] args) {
        Alpha main = new Alpha(args);
        main.main();
    }
}
```

And the commands:

```
javac Test.java
java Test 1 2
```

What is the result?

A(ans)
1525

B
13

C
Compilation fails

D
An exception is thrown at runtime

E
The program fails to execute due to runtime error

題解

題目執行的指令會編譯「Test.java」，由於 Test 類別有用 public 修飾，因此預設會執行 Test 類別的 main 方法，並把「1 2」作為 main 的參數。參數的索引值從 0 開始。

程式第 19 行，實體化 Alpha 物件會呼叫第 5 行的建構子，並傳入從命令列傳入的參數。在建構子中，會把傳入的字串陣列參數裡所有的元素都「串接」上「5」之後存進 Alpha 物件的 main 物件變數陣列對應的位置中。因此第 12 行程式會輸出「1525」。

(84)

Given the code fragment:

```
double discount = 0;
int qty = Integer.parseInt(args[0]);
// line n1;
```

And given the requirements:

If the value of the qty variable is greater than or equal to 90, discount = 0.5
If the value of the qty variable is between 80 and 90, discount = 0.2

Which two code fragments can be independently placed at line n1 to meet the requirements?

A(ans)

```
if (qty >= 90) {discount = 0.5; }
if (qty > 80 && qty < 90) {discount = 0.2; }
```

B

```
discount = (qty >= 90) ? 0.5 : 0;
discount = (qty > 80) ? 0.2 : 0;
```

C(ans)

```
discount = (qty >= 90) ? 0.5 : (qty > 80) ? 0.2 : 0;
```

D

```
if (qty > 80 && qty < 90) {
    discount = 0.2;
} else {
    discount = 0;
}
if (qty >= 90) {
    discount = 0.5;
} else {
    discount = 0;
}
```

E
```
discount = (qty > 80) ? 0.2 : (qty >= 90) ? 0.5 : 0;
```
題解
選項 A，程式邏輯符合題義。
選項 B，qty 若大於等於 90，那它一定會大於 80，因此第 7 行程式沒有作用，會被第 8 行的結果覆蓋。
選項 C，程式邏輯符合題義。
選項 D，qry 若介於 80 到 90 之間，discount 會在第二個 if 結構中，數值被改成 0。
選項 E，qty 若大於等於 90，那它一定會大於 80，因此第二個三元條件運算子條件永遠不會成立。

(85)
```
LocalDate date1 = LocalDate.now();
LocalDate date2 = LocalDate.of(2014, 6, 20);
LocalDate date3 = LocalDate.parse("2014-06-20",
DateTimeFormatter.ISO_DATE);
System.out.println("date1 = " + date1);
System.out.println("date2 = " + date2);
System.out.println("date3 = " + date3);
```

Assume that the system date is June 20, 2014. What is the result?

A
```
date1 = 2014-06-20
date2 = 2014-06-20
date3 = 2014-06-20
```

B
```
date1 = 2014-06-20
date2 = 2014-06-20
date3 = 2014-06-20
```

C
Compilation fails.

D
A DateParseException is thrown at runtime.
題解
這題是在考 Java 8 加入的日期與時間(Date-Time)API
now、of、parse 都是建立 LocalDate 物件的方式。parse 可以將字串使用指定的 DateTimeFormatter 格式化成日期與時間的物件，在這題程式中要格式化的字串是「2014-06-20」，符合 ISO_DATE 這個 DateTimeFormatter 的格式。

LocalDate 預設的 toString 方法會將日期以 ISO-8601 標準格式轉成字串。

(86)

```
class Mid {

    public int findMid(int n1, int n2) {
        return (n1 + n2) / 2;
    }
}

public class Calc extends Mid {

    public static void main(String[] args) {
        int n1 = 22, n2 = 2;
        // insert code here
        System.out.print(n3);
    }
}
```

Which two code fragments, when inserted at // insert code here, enable the code to compile and print 12?

A(ans)

```
Calc c = new Calc();
int n3 = c.findMid(n1, n2);
```

B

```
int n3 = super.findMid(n1, n3);
```

C

```
Calc c = new Mid();
int n3 = c.findMid(n1, n2);
```

D(ans)

```
Mid m1 = new Calc();
int n3 = m1.findMid(n1, n2);
```

E

```
int n3 = Calc.findMid(n1, n2);
```

題解

findMid 是物件方法，必須要實體化出 Calc 或是 Mid 的物件才能使用，因此選項 B、E 是錯的。

Calc 類別繼承 Mid 類別，所以 Calc 物件型態可以隱含式(implicit)轉型為 Mid 物件型態，反之則不行，因此選項 C 是錯誤的。

最後剩下的選項 A、D 是正確的。

(87)

```
public class String1 {

    public static void main(String[] args) {
        String s = "123";
        if (s.length() > 2) {
            s.concat("456");
        }
        for (int x = 0; x < 3; x++) {
```

```
            s += "x";
        }
        System.out.println(s);
    }
}
```

What is the result?

A
123

B(ans)
123xxx

C
123456

D
123456xxx

E
Compilation fails

題解
程式第 6 行，使用字串物件的「concat」方法會回傳字串連接後的結果，但並不會改變物件本身所表示的字串。因此這題只會在字串「123」後面串上 3 個「x」。

(88)

```
public class ColorTest {

    public static void main(String[] args) {
        String[] colors = {"red", "blue", "green", "yellow",
"maroon", "cyan"};
        int count = 0;
        for (String c : colors) {
            if (count >= 4) {
                break;
            } else {
                continue;
            }
            if (c.length() >= 4) {
                colors[count] = c.substring(0, 3);
            }
            count++;
        }
        System.out.println(colors[count]);
    }
}
```

What is the result?

A
Yellow

B
Maroon

C(ans)
Compilation fails

D
A StringIndexOutOfBoundsException is thrown at runtime.
題解
第 12 行會編譯錯誤，原因在於第 7 行的 if 不管條件式有沒有成立，都會直接略過之後 for 迴圈的程式，直接跳出迴圈或是進入下一次的迴圈，所以第 12 行的程式永遠執行不到，在編譯階段時就會被檢查出來。

(89)

```
public class Triangle {

    static double area;
    int b = 2, h = 3;

    public static void main(String[] args) {
        double p, b, h; // line n1
        if (area == 0) {
            b = 3;
            h = 4;
            p = 0.5;
        }
        area = p * b * h; // line n2
        System.out.println("Area is " + area);
    }
}
```

What is the result?

A
Area is 6.0

B
Area is 3.0

C
Compilation fails at line n1.

D(ans)
Compilation fails at line n2.
題解
由於第 7 行的 p、b、h 變數是在第 8 行的 if 內才初始化，然而 if 的程式可能會沒有被執行到，p、b、h 變數也就可能沒有被初始化，因此 line n2 會發生編譯錯誤。

(89)
Given the code fragment:

```
String[] colors = {"red", "blue", "green", "yellow", "maroon",
 "cyan"};
```

Which code fragment prints blue, cyan, ?

A(ans)

```
for (String c : colors) {
```

```
    if (c.length() != 4) {
        continue;
    }
    System.out.print(c + ", ");
}
```

B

```
for (String c : colors[]) {
    if (c.length() <= 4) {
        continue;
    }
    System.out.print(c + ", ");
}
```

C

```
for (String c : String[] colors) {
    if (c.length() >= 4) {
        continue;
    }
    System.out.print(c + ", ");
}
```

D

```
for (String c : colors) {
    if (c.length() >= 4) {
        System.out.print(c + ", ");
        continue;
    }
}
```

題解

選項 B、C 是錯誤的 foreach 用法，會編譯錯誤。

選項 A，會把所有字串長度等於 4 的元素輸出，是正確答案。

選項 D，會把所有字串長度大於等於 4 的元素輸出，不符合題目要求。

(90)

```
public class Test {

    public static void main(String[] args) {
        Test ts = new Test();
        System.out.print(isAvailable + " ");
        isAvailable = ts.doStuff();
        System.out.println(isAvailable);
    }

    public static boolean doStuff() {
        return !isAvailable;
    }
    static boolean isAvailable = false;
}
```

What is the result?

A

```
true true
```

B

true false

C(ans)
false true

D
false false

E
Compilation fails

題解
程式第 10 行的 doStuff 方法，會回傳 isAvailable 變數的相反邏輯運算結果。
第 5 行會輸出「false 」。第 6 行會把 isAvailable 變數改為 true。第 7 行會輸出「true」。

(91)
```
if (aVar++ < 10) {
    System.out.println(aVar + " Hello World!");
} else {
    System.out.println(aVar + " Hello Universe!");
}
```

What is the result if the integer aVar is 9?

A(ans)
10 Hello World!

B
Hello Universe

C
Hello World!

D
Compilation fails.

題解
「++」運算子若放置於變數後面，會先取值才會進行變數值再加一的動作，因此這邊的 if 條件式會成立，而輸出「10 Hello World!」。

(92)
```
public class SuperTest {

    public static void main(String[] args) {
        // statement1
        // statement2
        // statement3
    }
}

class Shape {

    public Shape() {
        System.out.println("Shape: constructor");
    }
```

```
    public void foo() {
        System.out.println("Shape: foo");
    }
}

class Square extends Shape {

    public Square() {
        super();
    }

    public Square(String label) {
        System.out.println("Square: constructor");
    }

    public void foo() {
        super.foo();
    }

    public void foo(String label) {
        System.out.println("Square: foo");
    }
}
```

What should statement1, statement2, and statement3, be respectively, in order to produce the result?

```
Shape: constructor
Square: foo
Shape: foo
```

A
```
Square square = new Square("bar");
square.foo("bar");
square.foo();
```

B
```
Square square = new Square("bar");
square.foo("bar");
square.foo("bar");
```

C
```
Square square = new Square();
square.foo();
square.foo(bar);
```

D
```
Square square = new Square();
square.foo();
square.foo("bar");
```

E
```
Square square = new Square();
square.foo();
square.foo();
```

F(ans)
```
Square square = new Square();
square.foo("bar");
square.foo();
```
題解
輸出的第 1 行為「Shape: constructor」，因此需要實體化出 Shape 物件才會執行第 13 行的程式。Square 類別繼承 Shape 類別，不傳入參數實體化出 Square 物件也會執行到 Shape 的建構子。所以目前看來，statement1 可以實體化出 Square 物件或是 Shape 物件，且選項 A、B 是錯誤的。
輸出的第 2 行為「Square: foo」，也就是要執行程式第 35 行，Square 類別內的 foo(String label)物件方法。所以 statement1 應該要實體化出 Square 物件，並且在 statement2 呼叫 Square 物件的 foo 方法時，必須傳入字串參數。因此選項 C、D、E 是錯誤的。
輸出的第 3 行為「Square: foo」，也就是要執行程式第 17 行，Shape 類別內的 foo()方法。這個方法在程式第 31 行，也就是 Square 類別內被覆寫，但覆寫的程式實作使用了「super」去呼叫了 Shape 類別原先的 foo()方法，所以呼叫 Square 物件的 foo()方法也會輸出「Square: foo」。選項 F 是正確答案。

(93)
```
default void print(){
}

Which statement is true?

A
This method is invalid.
```

B(ans)
This method can be used only in an interface.

```
C
This method can return anything.

D
This method can be used only in an interface or an
abstract class.

E
None of above.
```
題解
Java 8 的介面(interface)可以加入已經實作好程式的預設方法(default method)和靜態方法(static)。預設方法使用「default」來修飾，只能給實體化出來的物件來呼叫使用；靜態方法使用「static」來修飾，只能給介面來呼叫使用

(94)
```
interface Pet {
}

class Dog implements Pet {
}

public class Beagle extends Dog {
}
```

Which three are valid?

A(ans)
```
Pet a = new Dog();
```

B
```
Pet b = new Pet();
```

C
```
Dog f = new Pet();
```

D(ans)
```
Dog d = new Beagle();
```

E(ans)
```
Pet e = new Beagle();
```

F
```
Beagle c = new Dog();
```
題解
題目提供的程式，繼承關係是：「Beagle 是 Dog」與「Dog 是 Pet」，所以「Beagle 是 Pet」。
選項 A，實體化 Dog 物件，並用 Pet 型態的物件變數儲存參考，因為「Dog 是 Pet」，所以這是可以的。
選項 B，無法實體化 Pet 物件，因為 Pet 是介面(interface)。
選項 C，無法實體化 Pet 物件，因為 Pet 是介面(interface)。
選項 D，實體化 Beagle 物件，並用 Dog 型態的物件變數儲存參考，因為「Beagle 是 Dog」，所以這是可以的。
選項 E，實體化 Beagle 物件，並用 Pet 型態的物件變數儲存參考，因為「Beagle 是 Pet」，所以這是可以的。
選項 F，實體化 Dog 物件，並用 Beagle 型態的物件變數儲存參考，這個不行，因為「Dog 不一定是 Beagle」。

(95)
Given the code format:
```
class DBConfiguration {

    String user;
    String password;
}
```

And,
```
public class DBHandler {
    DBConfiguration configureDB(String uname, String password)
{
        // insert code here
    }
    public static void main(String[] args) {
        DBHandler r = new DBHandler();
        DBConfiguration   dbConf   =   r.configureDB("manager",
"manager");
    }
}
```

Which code fragment must be inserted at line 6 to enable the code to compile?

A
```
DBConfiguration f;
return f;
```

B
```
return DBConfiguration;
```

C(ans)
```
return new DBConfiguration();
```

D
```
retutn 0;
```

題解
程式第 5 行的 configureDB 方法要回傳 DBConfiguration 物件，因此第 6 行需要實體化出 DBConfiguration 物件並將結果回傳，才可以順利完成編譯。
選項 A，f 為物件參考變數，需要初始化後才可以使用。
選項 B，無法直接回傳一個類別。
選項 C，正確答案。
選項 D，回傳型態不能是數值 0，而是 DBConfiguration 物件才對。

(96)
Given:
```
public class Test {

    public static void main(String[] args) {
        Integer num = Integer.parseInt(args[1]);
        System.out.println("Number is : " + num);
    }
}
```

And the commands:
```
javac Test.java
java Test 12345
```

What is the result?

A
```
Number us : 12345
```

B
```
A NullPointerException is thrown at runtime
```

C
```
A NumberFormatException is thrown at runtime
```

D(ans)
An ArrayIndexOutOfBoundException is thrown at runtime.
題解
題目執行的指令會編譯「Test.java」，由於 Test 類別有用 public 修飾，因此預設會執行 Test 類別的 main 方法，並把「12345」作為 main 的參數。參數的索引值從 0 開始。

程式第 4 行要取得參數索引 1 的值，但是這裡的參數數量只有一個，因此會超出陣列索引範圍，而拋出 ArrayIndexOutOfBoundException 例外。

(97)

```
public class Series {

    private boolean flag;

    public void displaySeries() {
        int num = 2;
        while (flag) {
            if (num % 7 == 0) {
                flag = false;
            }
            System.out.print(num);
            num += 2;
        }
    }

    public static void main(String[] args) {
        new Series().displaySeries();
    }
}
```

What is the result?

A
2 4 6 8 10 12

B
2 4 6 8 10 12 14

C
Compilation fails

D
The program prints multiple of 2 infinite times

E(ans)
The program prints nothing
題解
由於 flag 變數是物件的欄位，因此它在宣告時會預設初始化為 false。第 7 行的 while 迴圈的執行條件永遠不會成立，程式也就不會輸出任何東西。

(98)
Which two are benefits of polymorphism?

A
Faster code at runtime

B
More efficient code at runtime

C(ans)
More dynamic code at runtime

D(ans)
More flexible and reusable code

E
Code that is protected from extension by other classes
題解
選項 A 及選項 B，多型並不會讓程式在執行的時候更快或是更有效率。
選項 C 和選項 D，多型可以使得程式更有彈性、重用性更高，執行時可以有更多的變化。
比如動物介面底下有人、狗、貓這三個類別，若有一個程式是呼叫動物介面的「走路」方法，那麼人、狗、貓等不同類別的實體可以有不同的動作。
選項 E，多型並不能保護程式被其他類別擴展，反倒可以幫助程式擴展。

(99)
```
public class StringReplace {

    public static void main(String[] args) {
        String message = "Hi everyone!";
        System.out.println("message = " + message.replace("e",
"X"));
    }
}
```

What is the result?.

A
message = Hi everyone!

B(ans)
message = Hi XvXryonX!

C
A compile time error is produced.

D
A runtime error is produced.

F
message = Hi Xveryone!
題解
字串物件的 replace 方法會將字串裡特定的字元或是子字串，全都取代成其它的字串。
因此：
Hi **e**v**e**ryon**e**! -> Hi **X**v**X**ryon**X**!
除了 replace 方法之外還有快速取代字串中符合正規表示式的子字串，所用的 replaceAll 方法以及 replaceFirst 方法。

(100)
```
public class TestScope {

    public static void main(String[] args) {
        int var1 = 200;
        System.out.print(doCalc(var1));
        System.out.print(" " + var1);
    }
```

```
    static int doCalc(int var1) {
        var1 = var1 * 2;
        return var1;
    }
}
```

What is the result?

A(ans)
400 200

B
200 200

C
400 400

D
Compilation fails.

題解
doCalc 方法會回傳傳入的 var1 參數值再乘以 2 的結果，但由於 Java 永遠都是「pass by value」，因此 main 方法的 var1 變數並不會被在 doCalc 方法內被改變。所以第 5 行會輸出「400」，第 6 行會輸出「 200」。

(101)
```
public class TestLoop {
    public static void main(String[] args) {
        float myarray[] = {10.20f, 20.30f, 30.40f, 50.60f};
        int index = 0;
        boolean isFound = false;
        float key = 30.40f;
        // insert code here
        System.out.println(isFound);
    }
}
```

Which code fragment, when inserted at line 7, enables the code print true?

A
```
while(key == myarray[index++]){
    isFound = true;
}
```

B
```
while(index <= 4){
    if(key == myarray[index]){
        index++;
        isFound = true;
        break;
    }
}
```

C
```
while(index++ < 5){
```

```
    if(key == myarray[index]){
        isFound = true;
    }
}
```

D(ans)

```
while(index < 5){
    if(key == myarray[index]){
        isFound = true;
        break;
    }
    index++;
}
```

題解

選項 A，while 迴圈會在 key 和 myarray 中 index 索引對應的值一樣才會執行。第一次進入 while 迴圈的時候，key = 30.40f、index = 0、myarray[0] = 10.20f，「30.40f」和「10.20f」並不相等，所以 while 迴圈不會執行，isFound 變數的值也就沒有機會被改為 true 了。

選項 B，while 迴圈會在 index 變數小於等於 4 的時候執行。第一次進入 while 迴圈的時候，index = 0，所以迴圈會執行，但因 30.40f 不等於 10.20f，所以 if 條件式不成立，繼續執行下一次迴圈。第二次進入 while 迴圈的時候，index 依然是 0，所以這個 while 迴圈是無窮迴圈。

選項 C，while 迴圈會在 index 變數小於等於 4 的時候執行，當條件式判斷後，會將 index 的值加 1。由於這個 while 沒有使用到 break 敘述，當 index=5 的時候，第 8 行會拋出 ArrayIndexOutOfBoundsException。

選項 D，正確答案，標準的線性搜尋演算法。

(101)

Which three statements are benefits of encapsulation?

A(ans)

Allows a class implementation to change without changing the clients

B(ans)

Protects confidential data from leaking out of the objects

C

Prevents code from causing exceptions

D(ans)

Enables the class implementation to protect its invariants

E

Permits classes to be combined into the same package

F

Enables multiple instances of the same class to be created safely

題解

選項 A，經過封裝過的類別可以保持 API 接口的一致，就算內部的程式實作有異動，也不需要去更改到其它引用了這個封裝好的類別的程式。舉例來說，某應用程式使用了其它封裝成 JAR 檔案的函式庫來進行影像處理的計算，而這個函式庫後來又有推出新版本來優化

計算的速度。此時若要更新這個應用程式，讓它去使用新版本的函式庫的話，只需要替換 JAR 檔案即可，不用更改或是重新編譯原先的應用程式。
選項 B，封裝可以隱藏程式碼，保護機密的資料外洩。
選項 C，封裝無法防止例外的發生。
選項 D，封裝可以保護類別的實作程式不被修改。
選項 E，封裝觀念和套件並沒有直接的關係。
選項 F，封裝與將一個類別實體化出不同物件的安全性並沒有關聯。

(102)

```
int row = 10;
for (; row > 0;) {
    int col = row;
    while (col >= 0) {
        System.out.print(col + " ");
        col -= 2;
    }
    row = row / col;
}
```

What is the result?

A(ans)
10 8 6 4 2 0

B
10 8 6 4 2

C
AnArithmeticException is thrown at runtime

D
The program goes into an infinite loop outputting: 10 8 6 4 2 0. . .

E
Compilation fails

題解
第 13 行的 for 迴圈會在 row 變數的值大於 0 的時候執行。
第一次執行 for 迴圈，row = 10、col = 10，所以 while 迴圈會重複執行，依序印出「10 8 6 4 」……，直到 col 變數的值小於 0。最後一次執行 while 迴圈時，col 的值為-2，小於 0 了，所以跳出 while 迴圈。
接著改變 row 的值為 10/-2=-5，然後準備進入下一次的 for 迴圈。然而，此時 row 小於等於 0 了，因此會直接跳出 for 迴圈。

(103)

```
public static void main(String[] args) {
    String theString = "Hello World";
    System.out.println(theString.charAt(11));
}
```

What is the result?

A
The program prints nothing

B
d

C(ans)
A StringIndexOutOfBoundsException is thrown at runtime.

D
An ArrayIndexOutOfBoundsException is thrown at runtime.

E
A NullPointerException is thrown at runtime.
題解
theString 變數所參考到的字串為長度是 11 的「Hello World」，程式第 7 行要取得「Hello World」索引位置為 11 的字元，會超出字串長度，因此會拋出 StringIndexOutOfBoundsException 例外。

(104)

```
public class Whizlabs {

    public static void main(String[] args) {
        LocalDate date = LocalDate.of(2015, 3, 26);
        Period p = Period.ofDays(1);
        System.out.println(date.plus(p));
    }
}
```

What is the output?

A(ans)
2015-03-27

B
2015-04-27

C
2015-02-27

D
Compilation fails due to error at line 6.

E
Compilation fails due to error at line 8.
題解
這題是在考 Java 8 加入的日期與時間(Date-Time)API
程式第 4 行，建立了一個西元 2015 年 3 月 26 日的 LocalDate 物件 date。
程式第 5 行，建立了一個一天的 Period 物件 p。
程式第 6 行，將 date 物件所代表的「西元 2015 年 3 月 26 日」加上週期物件 p 所代表的「一天」，會回傳代表「西元 2015 年 3 月 27 日」的 LocalDate 物件。

(105)

```
int[] lst = {1, 2, 3, 4, 5, 4, 3, 2, 1};
int sum = 0;
for (int frnt = 0, rear = lst.length - 1; frnt < 5 && rear >=
5; frnt++, rear--) {
```

```
    sum = sum + lst[frnt] + lst[rear];
}
System.out.print(sum);
```

What is the result?

A(ans)
20

B
25

C
29

D
Compilation fails

E
An ArrayIndexOutOfBoundsException is thrown at runtime

題解
程式第 8 行的 for 迴圈使用了兩種計次變數，分別是 frnt 和 rear。frnt 的範圍在 0~4，rear 的範圍在 8~5。
迴圈第一次執行，frnt = 0、rear = 8，sum = 0 + 1 + 1 = 2。
迴圈第二次執行，frnt = 1、rear = 7，sum = 2 + 2 + 2 = 6。
迴圈第三次執行，frnt = 2、rear = 6，sum = 6 + 3 + 3 = 12。
迴圈第四次執行，frnt = 3、rear = 5，sum = 12 + 4 + 4 = 20。
迴圈正要執行第五次時，frnt = 4、rear = 4，rear 變數已經小於 5 了，因此跳出迴圈，輸出 sum 變數的值。

(106)
```
class Base {

    public static void main(String[] args) {
        System.out.println("Base " + args[2]);
    }
}

public class Sub extends Base {

    public static void main(String[] args) {
        System.out.println("Overriden " + args[1]);
    }
}
```

And the commands:
```
javac Sub.java
java Sub 10 20 30
```

What is the result?

A
Base 30

B(ans)
Overridden 20

C
Overridden 20
Base 30

D
Base 30
Overridden 20

題解

題目執行的指令會編譯「Sub.java」，由於 Sub 類別有用 public 修飾，因此預設會執行 Sub 類別的 main 方法，並把「10 20 30」作為 main 的參數。
所以這題只會執行第 10 行的 main 方法，並且輸出「Overridden 20」。

(107)
Which three statements are true about the structure of a Java class?

A(ans)
A class can have only one private constructor.

B(ans)
A method can have the same name as a field.

C(ans)
A class can have overloaded static methods.

D
A public class must have a main method.

E
The methods are mandatory components of a class.

F
The fields need not be initialized before use.

題解

選項 A，一個類別可以擁有一個 private 修飾的私有建構子。這是對的，Java 並沒有限制私有建構子的數量，舉例來說：

```
public class Test {

    private Test(){

    }

    private Test(int a){

    }

    private Test(String s){

    }
}
```

選項 B，一個方法的名稱可以和欄位名稱相同。這是對的，舉例來說：

```
public class Test {

    int A;

    int A(){
        return A;
    }
}
```

選項 C，一個類別可以擁有多載(overload)的靜態類別。這是對的，舉例來說：

```
public class Test {

    public static void test(String a, String b){

    }

    public static void test(int a, int b){

    }
}
```

選項 D，一個公開的類別並不一定要有 main 方法。
選項 E，並不是每個類別都需要撰寫方法，甚至類別內什麼都不寫也可以。
選項 F，欄位並不是不用初始化，而是在宣告的時候就已經初始化了。

(108)

```
public class Test {

    public static void main(String[] args) {
        int day = 1;
        switch (day) {
            case "7":
                System.out.print("Uranus");
            case "6":
                System.out.print("Saturn");
            case "1":
                System.out.print("Mercury");
            case "2":
                System.out.print("Venus");
            case "3":
                System.out.print("Earth");
            case "4":
                System.out.print("Mars");
            case "5":
                System.out.print("Jupiter");
        }
    }
}
```

Which two modifications, made independently, enable the code to compile and run?

A
Adding a break statement after each print statement

B

Adding a default section within the switch code-block

C(ans)
Changing the string literals in each case label to integer

D(ans)
Changing the type of the variable day to String

E
Arranging the case labels in ascending order
題解
現有的程式會因為 switch 選擇的整數資料型態和 case 的字串資料型態不同而發生編譯錯誤。
選項 A，在每個 case 的 print 敘述下加入 break 敘述並無法修正編譯錯誤的問題。
選項 B，加入 default 選擇也無濟於事。
選項 C，將每個 case 所用的字串改成整數，這樣才可以符合 switch 的資料型態。
選項 D，將變數 day 的型態改成字串也可以符合 switch 的資料型態。
選項 E，case 排序的順序並不會影響到編譯結果。

(109)
```
float var1 = (12_345.01 >= 123_45.00) ? 12_456 : 14_56.02f;
float var2 = var1 + 1024;
System.out.print(var2);
```

What is the result?

A(ans)
13480.0

B
13480.02

C
Compilation fails

D
An exception is thrown at runtime
題解
先看到第 24 行的判斷式「12_345.01 >= 123_45.00」，這很明顯會成立，因此 var1 的值為「12456」。var2 的值為 12456+1024=13480。

(110)
```
public class App {

    public static void main(String[] args) {
        int i = 10;
        int j = 20;
        int k = j += i / 5;
        System.out.print(i + " : " + j + " : " + k);
    }
}
```

What is the result?

A
10 : 22 : 20

B(ans)
10 : 22 : 22

C
10 : 22 : 6

D
10 : 30 : 6
題解
「j += i / 5」會將 i 變數儲存的 10 先除以 5，再把這個值加回 j 變數，然後再回傳最後 j 變數的值。
(111)

```
class Caller {

    private void init() {
        System.out.println("Initialized");
    }

    public void start() {
        init();
        System.out.println("Started");
    }
}

public class TestCall {

    public static void main(String[] args) {
        Caller c = new Caller();
        c.start();
        c.init();
    }
}
```

What is the result?

A
Initialized
Started

B
Initialized
Started
Initialized

C(ans)
Compilation fails

D
An exception is thrown at runtime
題解

程式第 18 行使用了 Caller 物件的 init 方法。由於 init 方法被 private 修飾字修飾，它的可見度只有在 Caller 這個類別之內，所以 TestCall 類別無法去呼叫到 Caller 物件的 init 方法，會在編譯時出現錯誤。

(112)

```
public static void main(String[] args) {
    int x = 5;
    while (isAvailable(x)) {
        System.out.print(x);

    }
}

public static boolean isAvailable(int x) {
    return x-- > 0 ? true : false;
}
```

Which modification enables the code to print 54321?

A
Replace line 6 with System.out.print(--x);

B(ans)
At line 7, insert x--;

C
Replace line 6 with --x; and, at line 7, insert
System.out.print(x);

D
Replace line 12 With return (x > 0) ? false : true;

題解
程式第 12 行，在 isAvailable 方法內使用了「x--」，並不會影響到 main 方法的 x 變數，因為 Java 永遠為「pass by value」。
選項 A，會使輸出變成「43210」。
選項 B，可以成功輸出「54321」。
選項 C，這樣的邏輯和選項 A 是一樣的。
選項 D，while 迴圈將不會執行。

(113)

```
interface Readable {

    public void readBook();

    public void setBookMark();
}

abstract class Book implements Readable { // line n1

    public void readBook() {
    }
    // line n2
}

class EBook extends Book{ // line n3
```

```
    public void readBook(){

    }

    // line n4
}
```

Which options enable the code to compile?

A
Replace the code fragment at line n1 with:
```
class Book implements Readable {
```

B
At line n2 insert:
```
public abstract void setBookMark();
```

C(ans)
Replace the code fragment at line n3 with:
```
abstract class EBook extends Book{
```

D(ans)
At line n4 insert:
```
public void setBookMark() { }
```
題解
在尚未修改任何程式的時候，EBook 類別會因為沒有實作出 Readable 介面的 setBookMark 方法而發生編譯錯誤。
選項 A，將抽象類別 Book 改成非抽象類別。如此一來 Book 類別一定要實作出 Readable 介面的所有方法，會造成編譯錯誤。
選項 B，在 Book 抽象類別內新增 setBookMark 抽象方法，在 EBook 類別內依然沒有將這個抽象方法實作出來，所以還是一樣會發生編譯錯誤。
選項 C，將 EBook 類別改成抽象類別，如此一來 EBook 類別內不需實作出 Readable 介面的所有方法也沒關係。
選項 D，在 EBook 內實作出 setBookMark 方法，使得 Readable 介面的所有方法都被實作完成。

(114)
Which usage represents a valid way of compiling java source file with the name "Main"?

A(ans)
javac Main.java

B
java Main.class

C
java Main.java

D
java Main
題解
編譯 Java 原始碼要使用 javac 這個編譯器，用法如下：
javac 「.java」檔案的完整路徑

因此選項 A 是正確的。

(115)

```
public class MarkList {

    int num;

    public static void graceMarks(MarkList obj4) {
        obj4.num += 10;
    }

    public static void main(String[] args) {
        MarkList obj1 = new MarkList();
        MarkList obj2 = obj1;
        MarkList obj3 = null;
        obj2.num = 60;
        graceMarks(obj2);
    }
}
```

How many MarkList instances are created in memory at runtime?

A(ans)
1

B
2

C
3

D
4

題解
要實體化出一個物件，通常都是使用 new 運算子，這邊只需要算算看 new 的數量即可。由於程式只有在第 10 行使用到 new 運算子，因此 MarkList 物件只有被實體化出來一個。

(116)

```
public class Test {

    public static void main(String[] args) {
        Test t = new Test();
        int[] arr = new int[10];
        arr = t.subArray(arr, 0, 2);
    }

    // insert code here
}
```

Which method can be inserted at line // insert code here to enable the code to compile?

A(ans)

```
public int[] subArray(int[] src, int start, int end) {
```

```
      return src;
}
```

```
B
public int subArray(int src, int start, int end) {
      return src;
}
```

```
C
public int[] subArray(int src, int start, int end) {
      return src;
}
```

```
D
public int subArray(int[] src, int start, int end) {
      return src;
}
```

題解

程式第 6 行使用到 t 變數所參考到的 Test 物件的 subArray 方法，但是現有的程式並未定義與實作出 subArray 方法。這裡並不需要考慮 subArray 方法的執行結果是否正確，題目只要求編譯成功而已。

在第 6 行 subArray 方法傳入的三個引數分別是整數陣列、整數和整數，而回傳型態是整數陣列，因此選項 A 是正確的。

(117)

```
public class Msg {

    public static String doMsg(char x) {
        return "Good Day!";
    }

    public static String doMsg(int y) {
        return "Good Luck!";
    }

    public static void main(String[] args) {
        char x = 8;
        int z = '8';
        System.out.println(doMsg(x));
        System.out.print(doMsg(z));
    }
}
```

What is the result?

A(ans)

```
Good Day!
Good Luck!
```

```
B
Good Day!
Good Day!
```

```
C
```

```
Good Luck!
Good Day!
```

```
D
Good Luck!
Good Luck!
```

```
E
Compilation fails
```

題解

第 12 行，將數值 8 指派給字元變數 x 來儲存。第 13 行，將字元「8」的字元值指派給整數 z 變數來儲存。

第 14 行由於 x 變數是字元型態，因此會呼叫第 3 行的 doMsg 多載方法，輸出「Good Day!」。

第 15 行由於 x 變數是整數型態，因此會呼叫第 7 行的 doMsg 多載方法，輸出「Good Luck!」。

(118)

```
class X {

    int x1, x2, x3;
}

class Y extends X {

    int y1;

    Y() {
        x1 = 1;
        x2 = 2;
        y1 = 10;
    }
}

class Z extends Y {

    int z1;

    Z() {
        x1 = 3;
        y1 = 20;
        z1 = 100;
    }
}

public class Test3 {

    public static void main(String[] args) {
        Z obj = new Z();
        System.out.println(obj.x3 + ", " + obj.y1 + ", " +
obj.z1);
    }
}
```

```
Which constructor initializes the variable x3?
```

A(ans)
Only the default constructor of class X

B
Only the no-argument constructor of class Y

C
Only the no-argument constructor of class Z

D
Only the default constructor of object class

題解

程式第 31 行實體畫出 Z 物件，並把實體化出來的物件參考給變數 obj 儲存。在實體化 Z 物件的時候，會先執行第 21 行 Z 類別的建構子，由於這個建構子並沒有使用「super」或是「this」來指定要先呼叫哪個建構子，因此預設會在編譯時添加「super();」至第一行中。Z 類別的父類別是 Y 類別，因此 Z 類別建構子呼叫「super()」會執行第 10 行 Y 類別的建構子。基於同樣的原因，Y 類別的建構子也會去呼叫 X 類別的建構子，X 類別沒有撰寫建構子，所以在編譯時期會自動加入預設的 X 類別之建構子：

```
public X(){
    super();
}
```

在這個實體化出 Z 物件的過程中，並沒有在執行建構子的時候去初始化 x3 變數的值，x3 整數變數在被宣告出來的時候，就會預設為 0，因此這題描述比較合理的是選項 A。

(119)

```
int x = 100;
int a = x++;
int b = ++x;
int c = x++;
int d = (a < b) ? (a < c) ? a : (b < c) ? b : c;
System.out.println(d);
```

What is the result?

A
100

B
101

C
102

D
103

E
Compilation fails

題解

宣告 d 變數的那行會編譯錯誤，原因在於「?:」條件三元運算子的使用方式是錯誤的。條件三元運算子的「?」和「:」必須是成對出現，有幾個「?」就有幾個「:」。

(120)

Which of the following can fill in the blank in this code to make it compile?

Exam.java

```
public class Exam {

    void method() {
    }
}
```

OCAJP.java

```
public class OCAJP extends Exam{
    _____ void method(){}
}
```

A
abstract

B(ans)
Final

C
private

D
Default

E
Int

題解

在 Java 中，子類別若要覆寫父類別的方法，其方法的可見度只能等同或是大於父類別的方法。因此在要覆寫的方法原本沒有使用可見度修飾字(default)時，覆寫時只能同樣不使用可見度修飾字或是可見度範圍更高的 protected 或是 public。但在這題目中，並沒有適合的可見度選項能用，因此應該要保留不使用可見度修飾字。
所以選項 B，使用讓方法不再能被覆寫的 final 修飾字是最好的答案。

(121)

```
public class Test {
    public static void main(String[] args) {
        boolean isChecked = false;
        int arry[] = {1, 3, 5, 7, 8, 9};
        int index = arry.length;
        while (~code1~) {
            if (arry[index - 1] % 2 == 0) {
                isChecked = true;
            }
            ~code2~
        }
        System.out.print(arry[index] + ", " + isChecked);
    }
}
```

Which set of changes enable the code to print 1, true?

A(ans)
Replacing with index > 0 and replacing with index--;

B(ans)
Replacing with index > 0 and replacing with --index;

C
Replacing with index > 5 and replacing with --index ;

D
Replacing with index and replacing with --index ;

題解
程式最後的 arry[index]需為 1；isChecked 需為 true。一開始 index 的值為 6。
選項 A，可以成功讓 index 變數在跳出 while 迴圈的時候數值為 0，並且修改 isChecked 變數的值為 true。
選項 B，理由同選項 A。
選項 C，while 迴圈只執行了一次，並讓 index 變數儲存的值修改成 5，isChecked 的值因為 arry[5] % 2 為 1，因此輸出結果會變成：
9, false
選項 D，index 並非是布林變數，因此無法直接作為 while 迴圈的條件式，會發生編譯錯誤。

(122)
```
public class App {
    // Insert code here
        System.out.print("Welcome to the world of Java");
    }
}
```

Which two code fragments, when inserted independently at line // Insert code here, enable the program to execute and print the welcome message on the screen?

A(ans)
static public void main (String[] args) {

B
static void main (String[] args) {

C
public static void Main (String[] args) {

D(ans)
public static void main (String[] args) {

E
public void main (String[] args) {

題解
題目給的程式缺乏程式進入點–「main」靜態方法。因此需要將其正確加入至程式中。
選項 A，使用 static 和 public 來修飾「main」方法，讓它成為靜態且公開的方法，這是正確的程式進入點的使用方式。
選項 B，沒有使用 public 來修飾「main」方法，無法作為程式進入點。程式雖然可以編譯，但會找不到可進入的「main」方法。
選項 C，作為程式進入點的「main」方法的名稱只能是「main」，不能是「Main」，否則不會被當作程式進入點。

選項 D，正確，理由同選項 A。
選項 E，沒有使用 static 來修飾「main」方法，無法作為程式進入點。程式雖然可以編譯，但會找不到可進入的「main」方法。

(123)

```
public class TestField {

    int x;
    int y;

    public void doStuff(int x, int y) {
        this.x = x;
        y = this.y;
    }

    public void display() {
        System.out.print(x + " " + y + " : ");
    }

    public static void main(String[] args) {
        TestField m1 = new TestField();
        m1.x = 100;
        m1.y = 200;
        TestField m2 = new TestField();
        m2.doStuff(m1.x, m1.y);
        m1.display();
        m2.display();
    }
}
```

What is the result?

A
100 200 : 100 200 :

B
100 0 : 100 0 :

C(ans)
100 200 : 100 0 :

D
100 0 : 100 200 :
題解
先看到程式第 8 行，「doStuff」方法內並沒有將參數 y 的值指派給物件的 y 變數，因此「doStuff」方法只會更動物件的 x 變數。
第 17~18 行，直接去設定 m1 物件的 x 和 y 這兩個物件變數。設定完成後 m1.x 為 100, m1.y 為 200。
第 20 行使用 m2 物件的「doStuff」方法只會更動到物件變數 x，所以設定完成後 m2.x 為 100, m2.y 為 0。
因此選項 C 是正確的。

(124)

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

```
    String ta = "A ";
    ta = ta.concat("B ");
    String tb = "C ";
    ta = ta.concat(tb);
    ta.replace('C', 'D');
    ta = ta.concat(tb);
    System.out.println(ta);
}
```

What is the result?

A
A B C D

B
A C D

C
A B C

D
A B D

E
A B D C

F(ans)
A B C C
題解
第 7 行執行完後，ta 的值為「A B 」。第 9 行執行完後，ta 的值為「A B C 」。第 10 行並不會更動到 ta 變數的內容。第 11 行執行完後，ta 的值為「A B C C 」。

(125)

```
public class Test {

    public static void main(String[] args) {
        // line n1
        switch (x) {
            case 1:
                System.out.println("One");
                break;
            case 2:
                System.out.println("Two");
                break;
        }
    }
}
```

Which three code fragments can be independently inserted at line nl to enable the code to print one?

A(ans)

```
Byte x = 1;
```

B(ans)

```
short x = 1;
```

C
```
String x = "1";
```

D
```
Long x = 1;
```

E
```
Double x = 1;
```

F(ans)
```
Integer x = new Integer("1");
```
題解
從 switch 內的 case 來看，可以發現用於 case 的值為整數數值，所以 x 變數為整數型態的變數。但題目選項有使用到 Wrapper 類別，這裡要注意到 Java 的整數數值預設為 int 型態，但它能夠自動向下轉型至 short 或是 byte。所以也可以自動轉型(auto wrapping)成 Integer、Short 或是 Byte 物件。
選項 A，數字「1」可自動轉型成 Byte 物件。
選項 B，數字「1」可自動轉型成 short 基本資料型態。
選項 C，這裡的 switch 並不是使用字串型態的變數。
選項 D，數字「1」不可以自動轉型成 Long 物件，可以改成「1l」或是「1L」來轉型。
選項 E，數字「1」不可以自動轉型成 Double 物件。
選項 F，使用字串「1」來產生 Integer 物件，這是可以的。

(126)
```
class Vehicle {

    String type = "4W";
    int maxSpeed = 100;

    Vehicle(String type, int maxSpeed) {
        this.type = type;
        this.maxSpeed = maxSpeed;
    }
}

class Car extends Vehicle {

    String trans;

    Car(String trans) { // line n1
        this.trans = trans;
    }

    Car(String type, int maxSpeed, String trans) {
        super(type, maxSpeed);
        this(trans); //line n2
    }
}
```

And given the code fragment:

```
public static void main(String[] args) {
    Car c1 = new Car("Auto");
```

```
    Car c2 = new Car("4W", 150, "Manual");
    System.out.println(c1.type + " " + c1.maxSpeed + " " +
c1.trans);
    System.out.println(c2.type + " " + c2.maxSpeed + " " +
c2.trans);
}
```

What is the result?

A
```
4W 100 Auto
4W 150 Manual
```

B
```
null 0 Auto
4W 150 Manual
```

C
Compilation fails only at line n1

D
Compilation fails only at line n2

E(ans)
Compilation fails at both line n1 and line n2
題解
line n1 的建構子有誤，由於其並沒有使用「super」或是「this」敘述來決定要先執行的建構子，因此預設會自動加入「super();」於建構子的第一行。但是 Car 所繼承的 Vehicle 類別並沒有「Vehicle()」這個建構子，因此會有編譯錯誤。
line n2 使用了「this」敘述，但不是在建構子的第一行呼叫，因此會編譯錯誤。

(127)
```
public static void main(String[] args){
    ArrayList<String> list = new ArrayList<>();

    list.add("SE");
    list.add("EE");
    list.add("ME");
    list.add("SE");
    list.add("EE");

    list.remove("SE");

    System.out.print("Values are : " + list);
}
```

What is the result?

A
```
Values are : [EE, ME]
```

B
```
Values are : [EE, EE, ME]
```

C

```
Values are : [EE, ME, EE]
```

D
```
Values are : [SE, EE, ME, EE]
```

E(ans)
Values are : [EE, ME, SE, EE]

題解

List 集合允許擁有多個邏輯相同的元素，remove 方法會從頭尋找第一個符合的元素，並將其移除，所以在這題中只會把索引 0 的「SE」字串移除。

(128)
```
import java.util.ArrayList;
import java.util.List;

public class Whizlabs{
    public static void main(String[] args){
        List<Integer> list = new ArrayList<>();
        list.add(21); list.add(15);
        list.add(30); list.add(11);
        list.add(2);
        //insert here
        System.out.println(list);
    }
}
```

Which inserted at line 11, will provide the following output?

[21, 15, 11]

A
```
list.removeIf(e > e % 2 != 0);
```

B(ans(
list.removeIf(e -> e % 2 == 0);

C
```
list.remove(e -> e % 2 == 0);
```

D
None of the above.

題解

選項 A，會移除整數集合中，所有不能整除 2 的項目。所以集合中的元素會只剩下「30」和「2」。
選項 B，會移除整數集合中，所有能整除 2 的項目。所以集合中的元素會剩下「21」、「15」和「11」。
選項 C，remove 方法的用法錯誤。
選項 D，由於選項 B 就是答案，因此錯誤。

(129)
Traveler.java
```
class Tours {

    public static void main(String[] args) {
        System.out.print("Happy Journey! " + args[1]);
```

```
    }
}

public class Traveler {

    public static void main(String[] args) {
        Tours.main(args);
    }
}
```

And the commands:

```
javac Traveler.java
java Traveler Java Duke
```

What is the result?

A(ans)
Happy Journey! Duke

B
Happy Journey! Java

C
An exception is thrown at runtime

D
The program fails to execute due to a runtime error

題解
程式第 11 行，呼叫 Tours 類別的 main 靜態方法，並把 main 的 args 參數作為引數繼續傳遞過去。在 CLI 介面下傳入給 Java 程式的參數索引值是從 0 開始的，題目的要傳給 Traveler 類別的參數是「Java Duke」，因此 args[0]="Java"，args[1]="Duke"。

(130)

```
String color = "teal";

switch (color) {
    case "Red":
        System.out.println("Found Red");
    case "Blue":
        System.out.println("Found Blue");
        break;
    case "Teal":
        System.out.println("Found Teal");
        break;
    default:
        System.out.println("Found Default");
}
```

What is the result?3

A
Found Red
Found Default

```
B
Found Teal

C
Found Red
Found Blue
Found Teal

D
Found Red
Found Blue
Found Teal
Found Default
```

E(ans)
Found Default
題解
switch 內可以使用「break」關鍵字來使程式立刻跳出 switch。在這題目中，要判斷的字串為「teal」，雖然有「Teal」的 case，但它們「T」的大小寫不符合，因此 switch 只會執行到第 21~22 行，輸出「Found Default」

(131)

```
public class FieldInit {

    char c;
    boolean b;
    float f;

    void printAll() {
        System.out.println("c = " + c);
        System.out.println("c = " + b);
        System.out.println("c = " + f);
    }

    public static void main(String[] args) {
        FieldInit f = new FieldInit();
        f.printAll();
    }
}
```

```
What is the result?

A
c = null
b = false
f = 0.0F

B
c = 0
b = false
f = 0.0f

C
c = null
b = true
f = 0.0
```

D(ans)
c =
b = false
f = 0.0
題解
類別或是物件的欄位，字元型態的初始值為「'\0'」(字元值為 0)，數值型態的初始值為「0」，布林型態的初始值為「false」，物件型態的初始值為「null」。

(132)
```
public static void main(final String[] args) throws Exception
{
    String[][] arr = {{"A", "B", "C"}, {"D", "E"}};
    for (int i = 0; i < arr.length; i++) {
        for (int j = 0; j < arr[i].length; j++) {
            System.out.print(arr[i][j] + " ");
            if (arr[i][j].equals("B")) {
                break;
            }
        }
        continue;
    }
}
```

What is the result?

A
A B C

B
A B C D E

C(ans)
A B D E

D
Compilation fails.
題解
程式第 10 行的 if 條件式成立的時候會跳出迴圈。當程式輸出「B」字串後，這個 if 條件式就會成立，然後跳出裡面的 for 迴圈，接著依然會繼續執行外面的 for 迴圈。因此在這裡只是「C」字串被跳過輸出了而已。

(133)
Which two are Java Exception classes?

A(ans)
SercurityException

B
DuplicatePathException

C(ans)
IllegalArgumentException

D

TooManyArgumentsException

題解

SercurityException 是一種 RuntimeException，為使用 SecurityManager 類別時會拋出的例外類型，這個在開發 Android 應用程式的時候可能還蠻常見的，因為 Android 系統本身有對程式的權限進行細部控制。

IllegalArgumentException 也是一種 RuntimeException，當傳入方法的參數範圍有誤的時候，常會拋出這種例外。例如：

```
Integer.parseInt(null);
```

以上程式會拋出 NumberFormatException，NumberFormatException 是 IllegalArgumentException 的其中一種。

(134)

```
class Alpha {

    int ns;
    static int s;

    Alpha(int ns) {
        if (s < ns) {
            s = ns;
            this.ns = ns;
        }
    }

    void doPrint() {
        System.out.println("ns = " + ns + " s = " + s);
    }
}

public class TestA {

    public static void main(String[] args) {
        Alpha ref1 = new Alpha(50);
        Alpha ref2 = new Alpha(125);
        Alpha ref3 = new Alpha(100);
        ref1.doPrint();
        ref2.doPrint();
        ref3.doPrint();
    }
}
```

What is the result?

A
ns = 50 s = 125
ns = 125 s = 125
ns = 100 s = 125

B(ans)
ns = 50 s = 125
ns = 125 s = 125
ns = 0 s = 125

C

```
ns = 50 s = 50
ns = 125 s = 125
ns = 100 s = 100
```

D

```
ns = 50 s = 50
ns = 125 s = 125
ns = 0 s = 125
```

題解

ns 是物件變數，s 是類別變數，s 在物件的變化會影響到其他的物件。在第 6 行的 Alpha 建構子中，會使用比原先的 s 更大的值來初始化 ns 和 s。main 方法並非從小到大將數值傳給 Alpha 的建構子並實體化出不同的物件，第 22 行傳入的 125 是最大值，之後第 23 行傳入建構子的 100 將會使得第 7 行的 if 條件式不成立，而沒有在建構子中被初始化的 ns 變數數值預設是 0。

因此答案是選項 B。

(135)

```
class X {

    public void mX() {
        System.out.println("Xm1");
    }
}

class Y extends X {

    public void mX() {
        System.out.println("Xm2");
    }

    public void mY() {
        System.out.println("Ym");
    }
}

public class Test {

    public static void main(String[] args) {
        X xRef = new Y();
        Y yRef = (Y)xRef;
        yRef.mY();
        xRef.mX();
    }
}
```

What is the result?

A(ans)

Ym
Xm2

B

```
Ym
Xm1
```

C
Compilation fails

D
A ClassCastException is thrown at runtime
題解
程式第 22 行，將 Y 物件實體的參考指派給 X 型態的變數儲存，由於 Y 類別繼承 X 類別，隱含(implicit)式的向上轉型是可以的。
程式第 23 行，將 X 型態的變數用顯性(explicit)的方式向下轉型成 Y 型態，X 變數所參考的物件為 Y 物件，因此這個轉型也是可以的。
再來第 24,25 行，執行 Y 物件於第 14 行的 mY 方法和第 10 行 mX 方法。因此分別印出「Ym」和「Xm2

(136)

```
int a = -10;
int b = 17;
int c = expression1;
int d = expression2;
c++;
d--;
System.out.print(c + ", " + d);
```

What could expression1 and expression2 be, respectively, in order to produce output -8, 16?

A
++a, --b

B(ans)
++a, b—

C
a++, --b

D
a++, b—
題解
這題可以先往回推算至第 12 行執行完畢，最後 c = -8 - 1 = -9，d = 16 + 1 = 17。
所以第 12 行，要直接把 b 的值指派給 d，因此選項 A 和選項 C 都不對。第 11 行，要把 a 的值加一然後指派給 c，因此選項 C 和選項 D 都不對。所以答案是選項 B。

(137)

```
public class Whizlabs {
    public static void main(String[] args) {
        StringBuilder sb = new StringBuilder("1Z0");
        sb.concat("-808");
        System.out.println(sb);
    }
}
```

What is the output?

A

1Z0

B
1Z0-808

C
An exception will be thrown.

D
Compilation fails due to error at line 3.

E(ans)
Compilation fails due to error at line 4.
題解
StringBuilder 物件並沒有「concat」方法，因此程式第 4 行會編譯錯誤。如果要串接字串應使用 StringBuilder 的「append」方法。

(138)

```
public class App {

    String myStr = "7007";

    public void doStuff(String str) {
        int myNum = 0;
        try {
            String myStr = str;
            myNum = Integer.parseInt(myStr);
        } catch (NumberFormatException ne) {
            System.err.println("Error");
        }
        System.out.println("myStr: " + myStr + ", myNum: " +
myNum);
    }

    public static void main(String[] args) {
        App obj = new App();
        obj.doStuff("9009");
    }
}
```

What is the result?

A
myStr: 9009, myNum: 9009

B
myStr: 7007, myNum: 7007

C(ans)
myStr: 7007, myNum: 9009

D
Compilation fails

題解
程式第 8 行的「myStr」變數會遮蔽(shadow)掉第 3 行同名的「myStr」物件變數，作用範圍只在第 8~9 行的程式區塊內。在第 9 行會將字串「9007」轉成整數，並把轉好的整數數值儲存在第 6 行宣告的 myNum 變數中，這部份是沒問題的。在程式執行到第 12 行之後，「myStr」所指的是第 3 行的「myStr」物件變數，所以此時「myStr」存的是字串「7007」，而「myNum」存的是整數「9009」。

(139)

```
public class Test {
    public static void main(String[] args) {
        StringBuilder sb1 = new StringBuilder();
        String str1 = sb1.toString();
        // insert code here
        System.out.print(str1 == str2);
    }
}
```

Which code fragment, when inserted at line 9, enables the code to print true?

A(ans)

```
String str2 = str1;
```

B

```
String str2 = new String(str1);
```

C

```
String str2 = sb1.toString();
```

D

```
String str2 = "Duke";
```

題解
第 10 行的「==」運算子的運算元的數值或是參考必須要一樣才會是 true。因此這題答案是選項 A

(140)

```
public class Test {
    public static void main(String[] args) {
        int ax = 10, az = 30;
        int aw = 1, ay = 1;
        try {
            aw = ax % 2;
            ay = az / aw;
        } catch (ArithmeticException e1) {
            System.out.println("Invalid Divisor");
        } catch (Exception e2) {
            aw = 1;
            System.out.println("Divisor Changed");
        }
        ay = az / aw; // Line 14
        System.out.println("Succesful Division " + ay);
    }
}
```

What is the result?

A
Invalid Divisor
Divisor Changed
Successful Division 30

B
Invalid Divisor
Successful Division 30

C(ans)
Invalid Divisor
Exception in thread "main" java.lang.ArithmeticException: / by zero
at test.Teagle.main(Teagle.java:14)

D
Invalid Divisor
Exception in thread "main" java.lang.ArithmeticException: / by zero
at test.Teagle.main(Teagle.java:14)
Successful Division 1

題解

第 6 行「aw」變數的值是 10 除以 2 的餘數，也就是 0。因此第 7 行把值為 0 的「aw」變數作為除數，會拋出 ArithmeticException，然後在第 8 行被 catch 接到。

接著程式執行到第 14 行，依然會拋出 ArithmeticException，但是這個 ArithmeticException 會直接由 main 方法往外拋出給執行緒。

(141)

```
public class MyFor1 {
    public static void main(String[] args) {
        int[] x = {6, 7, 8};
        for (int i : x) {
            System.out.print(i + " ");
            i++;
        }
    }
}
```

What is the result?

A(ans)
6 7 8

B
7 8 9

C
0 1 2

D
6 8 10

E
Compilation fails

題解
這題只是將陣列內容使用第 4 行的 foreach 逐一將陣列元素輸出出來而已。第 6 行的「i++」是在第 5 行輸出「i」之後才執行，因此不會影響到我們陣列的輸出結果。

(142)

```
public class Product {
    int id;
    String name;
    public Product(int id, String name) {
        this.id = id;
        this.name = name;
    }
}
```

And given the code fragment:

```
public static void main(String[] args) {
    Product p1 = new Product(101, "Pen");
    Product p2 = new Product(101, "Pen");
    Product p3 = p1;
    boolean ans1 = p1 == p2;
    boolean ans2 = p1.name.equals(p2.name);
    System.out.print(ans1 + ":" + ans2);
}
```

What is the result?

A
true:true

B
true:false

C(ans)
false:true

D
false:false
題解
程式第 15 行的 ans1 值，是用「==」運算子來判斷 p1 和 p2 所儲存的物件參考是否相同，由於這兩者是不同的物件，因此會是 false。
程式第 16 行的 ans2 值，是用每個物件都有的「equals」方法來判斷 p1 和 p2 的「name」物件變數是否在邏輯上是一樣的，由於 p1 和 p2 的「name」變數所參考到的字串皆為「Pen」，因此會是 true。

(143)

```
String shirts[][] = new String[2][2];
shirts[0][0] = "red";
shirts[0][1] = "blue";
shirts[1][0] = "small";
shirts[1][1] = "medium";
```

Which code fragment prints red:blue:small:medium:?

A

```
for (int index = 1; index < 2; index++) {
```

```
    for (int idx = 1; idx < 2; idx++) {
        System.out.print(shirts[index][idx] + ":");
    }
}
```

```
B
for (int index = 0; index < 2; index++) {
    for (int idx = 0; idx < index; idx++) {
        System.out.print(shirts[index][idx] + ":");
    }
}
```

```
C
for (String sizes : shirts) {
    for (String s : sizes) {
        System.out.print(s + ":");
    }
}
```

D(ans)

```
for (int index = 0; index < 2;) {
    for (int idx = 0; idx < 2;) {
        System.out.print(shirts[index][idx] + ":");
        idx++;
    }
    index++;
}
```

題解

選項 A，由於 index 和 idx 都是從 1 開始，所以只會輸出 shirts[1][1]的內容，也就是「medium」字串還有「:」。

選項 B，會輸出「small:」。

選項 C，shirt 的型態應為字串陣列「String[]」。

選項 D，正確答案。

(144)

```
class A {

    public A() {
        System.out.print("A ");
    }
}

class B extends A {

    public B() {
        System.out.print("B ");
    }
}

class C extends B {

    public C() {
        System.out.print("C ");
    }
```

```
    public static void main(String[] args) {
        C c = new C();
    }
}
```

What is the result?

A
C B A

B
C

C(ans)
A B C

D
Compilation fails at line n1 and line n2

題解

程式第 22 行，實體化出了 C 的物件，因此會執行 C 的建構子。
建構子若並沒有使用「super」或是「this」來定義要先執行哪個建構子，在編譯階段時會自動在第一行加上「super();」。因此程式其實可以擴展成這樣：

```
class A {

    public A() {
        super();
        System.out.print("A ");
    }
}

class B extends A {

    public B() {
        super();
        System.out.print("B ");
    }
}

class C extends B {

    public C() {
        super();
        System.out.print("C ");
    }

    public static void main(String[] args) {
        C c = new C();
    }
}
```

所以會依序輸出「A 」、「B 」、「C 」

(145)

```
abstract class A1 {

    public abstract void m1();
```

```
    public void m2() {
        System.out.println("Green");
    }
}

abstract class A2 extends A1 {

    public abstract void m3();

    public void m1() {
        System.out.println("Cyan");
    }

    public void m2() {
        System.out.println("Blue");
    }
}

public class A3 extends A2 {

    public void m1() {
        System.out.println("Yellow");
    }

    public void m2() {
        System.out.println("Pink");
    }

    public void m3() {
        System.out.println("Red");
    }

    public static void main(String[] args) {
        A2 tp = new A3();
        tp.m1();
        tp.m2();
        tp.m3();
    }
}
```

What is the result?

A(ans)
Yellow
Pink
Red

B
Cyan
Blue
Red

C

Cyan
Green
Red

D
Compilation Fails

題解

tp 變數所參考到的是 A3 類別的物件實體，A3 類別覆寫了 m1, m2, m3 方法。
程式第 39 行會執行 A3 類別覆寫的 m1 方法，輸出「Yellow」。
程式第 40 行會執行 A3 類別覆寫的 m2 方法，輸出「Pink」。
程式第 41 行會執行 A3 類別覆寫的 m2 方法，輸出「Red」。

(146)

```
public class Test {

    static void dispResult(int[] num) {
        try {
            System.out.println(num[1] / (num[1] - num[2]));
        } catch (ArithmeticException e) {
            System.err.println("First exception");
        }
        System.out.println("Done");
    }

    public static void main(String[] args) {
        try {
            int[] arr = {100, 100};
            dispResult(arr);
        } catch (IllegalArgumentException e) {
            System.err.println("Second exception");
        } catch (Exception e) {
            System.err.println("Third exception");
        }
    }
}
```

What is the result?

A
0
Done

B
First Exception
Done

C
Second Exception

D
Done
Third Exception

E(ans)
Third Exception

題解
程式第 5 行會因為 num 所參考到的陣列物件並沒有索引 2 這個位置，而拋出 ArrayIndexOutOfBoundsException。而這個 ArrayIndexOutOfBoundsException 並不是 ArithmeticException，所以不會被第 6 行的 catch 接到，而由「dispResult」方法繼續向外拋出，最終被第 18 行的 catch 接到，輸出「Third Exception」。

(147)

```
class Test2 {

    int fvar;
    static int cvar;

    public static void main(String[] args) {
        Test2 t = new Test2();
        // insert code here to write field variables
    }
}
```

Which code fragments, inserted independently, enable the code compile?

A(ans)
t.fvar = 200;

B(ans)
cvar = 400;

C
fvar = 200;
cvar = 400;

D
this.fvar = 200;
this.cvar = 400;

E(ans)
t.fvar = 200;
Test2.cvar = 400;

F
this.fvar = 200;
Test2.cvar = 400;

題解
選項 A，用物件實體的參考變數去存取物件實體的欄位，這個正確。
選項 B，直接在靜態方法「main」裡面使用靜態的欄位，這個也正確。
選項 C，直接在靜態方法「main」裡存取物件實體的欄位，因為沒有物件，所以無法存取。
選項 D，直接在靜態方法「main」裡使用「this」來表示目前的物件實體，因為沒有物件，所以無法存取。
選項 E，用物件實體的參考變數去存取物件實體的欄位，和用類別名稱去存取類別欄位，這個也正確。
選項 F，不能使用「this」，理由同選項 D。

(148)

```
public class ScopeTest {

    int j;
    int k;

    public static void main(String[] args) {
        new ScopeTest().doStuff();
    }

    void doStuff() {
        int x = 5;
        doStuff2();
        System.out.println("x");
    }

    void doStuff2() {
        int y = 7;
        System.out.println("y");
        for (int z = 0; z < 5; z++) {
            System.out.println("z");
            System.out.println("y");
        }
    }
}
```

Which two items are fields?

A(ans)
j

B(ans)
k

C
x

D
y

E
Z

題解

這個題目是在問，哪兩個變數是欄位。欄位就是類別或是物件變數。因此只有選項 A 的「j」和選項 B 的「k」符合。

(149)

```
public class Test {

    void readCard() throws Exception {
        System.out.println("Reading Card");
    }

    void checkCard(int carNo) throws Exception {
        System.out.println("Checking Card");
    }
```

```
    public static void main(String[] args) throws RuntimeException { //
line n1
        Test ex = new Test();
        int carNo = 12344;
        ex.checkCard(carNo); // line n2
        ex.readCard(carNo); // line n3
    }
}
```

What is the result?

A
Reading Card
Checking Card

B
Compilation fails only at line n1.

C
Compilation fails only at line n2.

D
Compilation fails only at line n3.

E(ans)
Compilation fails at both line n2 and line n3.
題解
「checkCard」方法使用 throws 關鍵字來將 Exception 拋出，Exception 是屬於 checked exception，因此在「main」方法中必須要撰寫程式來處理「checkCard」方法所拋出的 Exception。「main」方法雖然有用 throws 將 RuntimeException 拋出，但是 Exception 並不是 RuntimeException(Exception 不是繼承 RuntimeException)，因此這題 line n2 和 line n3 會因為沒有處理 Exception 而發生編譯錯誤。

(150)

```
public class Test {

    public static void main(String[] args) {
        String[][] chs = new String[2][];
        chs[0] = new String[2];
        chs[1] = new String[5];
        int i = 97;

        for (int a = 0; a < chs.length; a++) {
            for (int b = 0; b < chs.length; b++) {
                chs[a][b] = "" + i;
                i++;
            }
        }

        for (String[] ca : chs) {
            for (String c : ca) {
                System.out.print(c + " ");
            }
            System.out.println();
```

```
        }
    }
}
```

What is the result?

A(ans)
97 98
99 100 null null null

B
91 98
99 100 101 102 103

C
Compilation fails.

D
A NullPointerException is thrown at runtime.

E
An ArrayIndexOutOfBoundsException is thrown at runtime.

題解
第 10 行的 for 迴圈一開始，a = 0, b = 0，所以 chs[0][0] = "97"。
接下來，a = 0, b = 1，所以 chs[0][1] = "98"。
然後，a = 1, b = 0，所以 chs[1][0] = "99"。
最後，a = 1, b = 1，所以 chs[1][1] = "100"，跳出迴圈。
其他沒設定到的陣列元素，初始值為 null。

(151)
```
public class TestA extends Root {

    public static void main(String[] args) {
        Root r = new TestA();
        System.out.println(r.method1()); // line n1
        System.out.println(r.method2()); // line n2
    }
}

class Root {

    private static final int MAX = 20000;

    private int method1() {
        int a = 100 + MAX;  // line n3
        return a;
    }

    protected int method2() {
        int a = 200 + MAX;  // line n4
        return a;
    }
}
```

Which line causes a compilation error?

A(ans)
Line n1

B
Line n2

C
Line n3

D
Line n4

題解
line n1 會編譯錯誤，因為「method1」方法是 Root 類別的私有成員，無法由外面的類別存取。

(152)
```
StringBuilder sb = new StringBuilder() ;
sb.append("world");
```

Which code fragment prints Hello world?

A(ans)
```
sb.insert(0,"Hello ");
System.out.println(sb);
```

B
```
sb.append(0,"Hello ");
System.out.println(sb);
```

C
```
sb.append(0,"Hello ");
System.out.println(sb);
```

D
```
sb.set(0,"Hello ");
System.out.println(sb);
```

題解
sb 變數所參考的 StringBuilder 物件一開始的字串內容是「world」，如果要輸出「Hello world」，就必須要將「Hello 」字串插入至「world」之前。
選項 A，使用 StringBuilder 物件的「insert」方法來插入「Hello 」字串至開頭的位置，這是正確的。
選項 B，使用 StringBuilder 物件的「append」方法會使得 StringBuilder 物件的內容變成「worldHello 」。
選項 C，StringBuilder 物件沒有「add」方法。
選項 D，StringBuilder 物件沒有「set」方法。

(153)
Foo.java
```
package facades;

public interface Foo { }
```

Boo.java

```
package facades;

public interface Boo extends Foo { }
```

Woofy.java

```
package org.domain;

// line n1

public class Woofy implements Boo, Foo { }
```

Test.java

```
package org;

// line n2

public class Test {
    public static void main(String[] args) {
        Foo obj = new Woofy();
    }
}
```

Which set modifications enable the code to compile and run?

A

At line n1, Insert:

```
import facades;
```

At line n2, insert:

```
import facades;
import org.domain;
```

B

At line n1, Insert:

```
import facades.*;
```

At line n2, insert:

```
import facades.*;
import org.*;
```

C

At line n1, Insert:

```
import facades.*;
```

At line n2, insert:

```
import facades.Boo;
import org.*;
```

D

At line n1, Insert:

```
import facades.Foo, Boo;
```

At line n2, insert:

```
import org.domain.Woofy;
```

E(ans)

At line n1, Insert:
import facades.*;

At line n2, insert:
import facades.*;
import org.domain.Woofy;
題解
import 關鍵字可以引入類別或是介面，不是引入套件(package)，因此選項 A 是錯誤的。
而選項 B 和選項 C，需注意到「*」所指的是該套件下的所有類別或是介面，並不包含子套件。實際上也沒有在分子、母套件，只要套件名稱不同就是不同的套件。
選項 D，語法錯誤了，不能使用「,」來分別引入不同的類別。
選項 E 才是正確的。

(154)
```
public static void main(String[] args) {
    List colors = new ArrayList();
    colors.add("green");
    colors.add("red");
    colors.add("blue");
    colors.add("yellow");
    colors.remove(2);
    colors.add(3, "cyan");
    System.out.print(colors);
}
```

What is the result?

A(ans)
[green, red, yellow, cyan]

B
[green, blue, yellow, cyan]

C
[green, red, cyan, yellow]

D
Am IndexOutOfBoundsException is thrown at runtime
題解
程式第 15 行的 remove 方法，將 colors 物件裡的「blue」字串移除了。此時 colors 這個 ArrayList 物件所存的元素為["green", "red", "yellow"]。
程式第 16 行的 add 方法，將「cyan」字串插入至索引 3 的位置，也就是「yellow」字串的下一個位置。此時 colors 這個 ArrayList 物件所存的元素為["green", "red", "yellow", "cyan"]。

(155)
```
public static void main(String[] args) {
    boolean opt = true;
    switch (opt) {
        case true:
            System.out.print("True");
            break;
        default:
```

```
            System.out.print("***");
        }
        System.out.println("Done");
}
```

Which modification enables the code fragment to print TrueDone?

A(ans)
Replace line 5 With String result = "true";
Replace line 7 with case "true":

B
Replace line 5 with boolean opt = 1;
Replace line 7 with case 1:

C
At line 9, remove the break statement.

D
Remove the default section.
題解
Java 的 switch 只能用來判斷數值、字元，或是字串(Java 7)之後，因此題目的程式使用 boolean 的方式是錯誤的，必須要將其修改成正確的才行。
選項 A，正確。
選項 B，boolean 的值不能是數字 1。
選項 C，拿掉 break 也無濟於事。因為 switch 本身的用法是錯的。
選項 D，拿掉 default 也無濟於事。因為 switch 本身的用法是錯的。

(156)
```
public class Test2 {
    public static void main(String[] args) {
        int ar1[] = {2, 4, 6, 8};
        int ar2[] = {1, 3, 5, 7, 9};
        ar2 = ar1;
        for (int e2 : ar2) {
            System.out.print(" " + e2);
        }
    }
}
```

What is the result?

A(ans)
2 4 6 8

B
2 4 6 8 9

C
1 3 5 7

D
1 3 5 7 9
題解

程式第 5 行，已經將原先 ar2 所參考到的陣列物件拋棄了，無需再顧慮它。如果這邊是要將原先 ar1 陣列物件的元素複製給 ar2 的話，應該要這樣寫：

```
public class Test2 {
    public static void main(String[] args) {
        int ar1[] = {2, 4, 6, 8};
        int ar2[] = {1, 3, 5, 7, 9};
        System.arraycopy(ar1, 0, ar2, 0, ar1.length);
        for (int e2 : ar2) {
            System.out.print(" " + e2);
        }
    }
}
```

(157)

```
public class Test {

    static boolean bVar;

    public static void main(String[] args) {

        boolean bVar1 = true;
        int count = 8;
        do {
            System.out.println("Hello Java! " + count);
            if (count >= 7) {
                bVar1 = false;
            }
        } while (bVar != bVar1 && count > 4);
        count -= 2;
    }
}
```

What is the result?

A
Hello Java! 8
Hello Java! 6
Hello Java! 4

B
Hello Java! 8
Hello Java! 6

C(ans)
Hello Java! 8

D
Compilation fails

題解

這題程式的邏輯還蠻混亂的，真正在寫程式的時候不會寫出這種爛 code。首先這邊要知道，類別變數 bVar 在沒有指定其初始值的狀況下，預設會是 false，所以根據第 14 行的 while 條件，可以知道當 bVar1 為 false 的時候，條件式就不成立了，迴圈也不會繼續執行。

再來看 count 這個變數，count 在 do-while 迴圈內並沒有做任何的更動，因此第 14 行的 while 條件中的 count 大於 4 是永遠成立的。

第 11 行的 if 在迴圈第一次執行的時候就會成立，因此會將 bVar1 設成 false，迴圈只執行一次就結束了。最後「Hello Java! 8」只輸出了一次。

(158)

Given the code fragment:

```
class Student{
    String name;
    int age;
}
```

And,

```
public class Test{
    public static void main(String[] args){
        Student s1 = new Student();
        Student s2 = new Student();
        Student s3 = new Student();
        s1 = s3;
        s3= s2;
        s2 = null;
    }
}
```

Which statement is true?

A
After line 8, three objects are eligible for garbage collection

B
After line 8, two objects are eligible for garbage collection

C(ans)
After line 8, one object is eligible for garbage collection

D
After line 8, none of the objects are eligible for garbage collection

題解

在這裡按照順序把實體化出來的 Student 物件標記為 A,B,C。程式執行完第 8 行時，A 物件已經沒有被任何變數參考，B 物件被 s3 變數參考，C 物件被 s1 變數參考。沒有被任何變數參考的 A 物件會在稍候被垃圾回收(Garbage Collection)。

(159)

```
public class Test {

    public static void main(String[] args) {
        int arr[] = new int[4];
        arr[0] = 1;
        arr[1] = 2;
        arr[2] = 4;
        arr[3] = 5;
        int sum = 0;
        try {
            for (int pos = 0; pos <= 4; pos++) {
                sum = sum + arr[pos];
```

```
            }
        } catch (Exception e) {
            System.out.println("Invalid index");
        }
        System.out.println(sum);
    }
}
```

What is the result?

A
12

B(ans)
Invalid Index
12

C
Invalid Index

D
Compilation fails

題解

程式第 11 行的 for 迴圈的 pos 變數的值為「0,1,2,3,4」，當 pos 變數數值為 4 的時候，第 12 行會因為陣列索引超出陣列長度，而拋出 ArrayIndexOutOfBoundsException，這個例外會在第 14 行被 catch 接住。因此程式會先輸出「Invalid index」，接著再輸出 1+2+4+5 的結果。

(160)

```
public class MainMethod {

    void main() {
        System.out.println("one");
    }

    static void main(String args) {
        System.out.println("two");
    }

    public static void main(String[] args) {
        System.out.println("three");
    }

    void mina(Object[] args) {
        System.out.println("four");
    }
}
```

What is printed out when the program is excuted?

A
one

B
two

C(ans)
three

D
four
題解
Java 程式的進入點在「main」方法。「main」方法位於 public 類別下，且使用 public 來修飾，傳入參數為字串陣列，可直接從命令列介面(CLI, Command Line Interface)接收參數。
標準的「main」方法寫法如下：

```
public static void main(String[] args) {
}
```

也可以使用 varargs：

```
public static void main(String... args) {
}
```

(161)

```
public class Greeting {

    public static void main(String[] args) {
        System.out.println("Hello " + args[0]);
    }
}
```

Which set of commands prints Hello Duke in the console?

A

```
javac Greeting
java Greeting Duke
```

B

```
javac Greeting.java Duke
java Greeting
```

C(ans)

```
javac Greeting.java
java Greeting Duke
```

D

```
javac Greeting.java
java Greeting.class Duke
```

題解
「javac」是 Java 的編譯器(Compiler)，可以用來編譯 Java 原始碼，也就是「.java」檔案，指令用法如下：
`javac 「.java」檔案的路徑`
「java」是用來執行 Java 程式的指令工具，可以用來執行 Java 的類別，也就是「.class」檔案，指令用法如下：
`java 套件與類別的名稱`
若要傳入參數給 main 方法，用法如下：
`java 套件與類別的名稱 [參數一 [參數二 [參數三 ...]]]`

(162)

```
import java.util.ArrayList;
```

```
import java.util.List;

public class Ref {

    public static void main(String[] args) {
        StringBuilder s1 = new StringBuilder("Hello Java!");
        String s2 = s1.toString();
        List<String> lst = new ArrayList<String>();
        lst.add(s2);
        System.out.println(s1.getClass());
        System.out.println(s2.getClass());
        System.out.println(lst.getClass());
    }
}
```

What is the result?

A
class java.lang.String
class java.lang.String
class java.util.ArrayList

B
class java.lang.Object
class java.lang. Object
class java.util.Collection

C(ans)
class java.lang.StringBuilder
class java.lang.String
class java.util.ArrayList

D
class java.lang.StringBuilder
class java.lang.String
class java.util.List

題解
這題只是要知道「getClass」看的是變數所參考到的物件實體是屬於哪個類別即可。

(163)
Which two statements are true for a two-dimensional array of primitive data type?

A(ans)
It cannot contain elements of different types.

B
The length of each dimension must be the same.

C
At the declaration time, the number of elements of the array in each dimension must be specified.

D(ans)
All methods of the class object may be invoked on the two-dimensional array.

題解
選項 A，基本資料型態的二維陣列不能包含其他不同型態的元素，這是正確的。
選項 B，Java 允許多維陣列每個維度擁有不同的長度。
選項 C，定義陣列型態的時候，可以先不用把每個維度的元素數量給決定好，實體化陣列物件的時候再決定即可。
選項 D，所有在 Object 類別裡的方法都可以在陣列中調用，因為陣列也是物件，同樣繼承自 Object 類別。

(164)

```
public class ForTest {

    public static void main(String[] args) {
        int[] array = {1, 2, 3};
        for (foo) {
        }
    }
}
```

Which three code fragments, when replaced individually for foo, enables the program to compile?

A(ans)
int i : array

B(ans)
int i = 0; i < 1;

C(ans)
; ;

D
; i < 1; i++ E. i = 0; i<1;

題解
選項 A 為 Java foreach 的正確用法。
選項 B 也是標準的 for loop 用法，只是因為少了步進欄位，i 變數的數值永遠不會被改變，所以會變成無窮迴圈。
選項 C 也是正確的無窮迴圈。
選項 D，變數 i 沒有事先宣告。
選項 E，理由同選項 D。

(165)

```
int b = 3;
if (!(b > 3)) {
    System.out.println("square ");
}{
    System.out.println("circle ");
}
System.out.println("...");
```

What is the result?

A
square
...

B
circle
...

C(ans)
square
circle
...

D
Compilation fails.

題解

b 的變數數值為 3，程式第 4 行的 if 條件式「b 不大於 3」成立，先輸出「square 」。接著注意第 6 行 if 區塊中止的位置，他並沒有使用 else 或是 else if 關鍵字，而直接使用了一個新的程式區塊，這樣的作法是可以被接受的，程式會繼續執行到第 7 行，輸出「circle 」。最後執行到第 9 行，輸出「...」。

(166)

```
public class TestOerator {

    public static void main(String[] args) {
        int result = 30 - 12 / (2 * 5) + 1;
        System.out.print("Result = " + result);
    }
}
```

What is the result?

A
Result = 2

B
Result = 3

C
Result = 28

D
Result = 29

E(ans)
Result = 30

題解

Java 的算式遵循「先乘除，後加減」和「括號內先計算」的規則，因此此題的運算過程如下：

result = 30 - 12 / (2 * 5) + 1 = 30 - **12 / 10** + 1 = 30 - **1** + 1 = 30

這裡還有一點需要注意到的地方是，Java 的整數除法運算，會自動省略小數點的部份。

(167)

```
class SpecialException extends Exception {

    public SpecialException(String message) {
        super(message);
        System.out.println(message);
```

```
    }
}

public class ExceptionTest {
    public static void main(String[] args) {
        try {
            doSomething();
        } catch (SpecialException e) {
            System.out.println(e);
        }
    }
    static void doSomething() throws SpecialException {
        int[] ages = new int[4];
        ages[4] = 17;
        doSomethingElse();
    }
    static void doSomethingElse() throws SpecialException {
        throw new SpecialException("Thrown at end of
doSomething() method");
    }
}
```

What will be the output?

A
SpecialException: Thrown at end of doSomething() method

B
Error in thread "main" java.lang.ArrayIndexOutOfBoundsError

C(ans)
Exception
in thread "main" java.lang.ArrayIndexOutOfBoundsException: 4
at ExceptionTest.doSomething(ExceptionTest.java:13)
at ExceptionTest.main(ExceptionTest.java:4)

D
SpecialException: Thrown at end of doSomething() method
at ExceptionTest.doSomethingElse(ExceptionTest.java:16)
at ExceptionTest.doSomething(ExceptionTest.java:13)
at ExceptionTest.main(ExceptionTest.java:4)

題解

程式會在第 19 行拋出 ArrayIndexOutOfBoundsException，再從 main 方法繼續向外拋出，導致執行緒產生例外中斷。因為 ArrayIndexOutOfBoundsException 並不是 SpecialException 的子類別，因此無法被第 13 行的 catch 接住。

(168)

```
public class Test {

    public static void main(String[] args) {
        try{
            String[] arr = new String[4];
            arr[1] = "Unix";
            arr[2] = "Linux";
            arr[3] = "Solarios";
            for(String var:arr){
```

```
                System.out.print(var + " ");
            }
        }catch(Exception e){
            System.out.print(e.getClass());
        }
    }
}
```

What is the result?

A
Unix Linux Solaris

B(ans)
null Unix Linux Solaris

C
Class java.lang.Exception

D
Class java.lang.NullPointerException

題解

第 8 行程式執行後，arr 字串陣列所參考到的陣列物件內容為「{null, "Unix", "Linux", "Solarios"}」，這裡比較會有問題的是索引 0 那個位置的 null。在 Java 中，若「+」運算子的運算元其中至少由一個是字串(物件)型態的話，會進行字串連接運算，null 在此時會自動轉成「null」字串。

(169)

```
class Patient {

    String name;

    public Patient(String name) {
        this.name = name;
    }
}

public class Test {
    public static void main(String[] args) {
        List ps = new ArrayList();
        Patient p2 = new Patient("Mike");
        ps.add(p2);

        // insert code here

        if (f >= 0) {
            System.out.print("Mike Found");
        }
    }
}
```

Which code fragment, when inserted at line 14, enables the code to print Mike Found?

A

```
int f = ps.indexOf{new Patient("Mike")};
```

B
```
int f = ps.indexOf(Patient("Mike"));
```

C
```
Patient p = new Patient("Mike");
int f = ps.indexOf(p);
```

D(ans)
```
int f = ps.indexOf(p2);
```
題解
選項 A，語法錯誤，方法的括號應為小括號，而非大括號。
選項 B，語法錯誤，少了 new 運算元。
選項 C，雖然 p 和 p2 變數所參考的物件其實體化的方式都相同，但它們是兩個獨立的物件。List 的「indexOf」方法會利用每個物件皆有的「equals」方法來判斷兩個物件參考是否儘管參考到的物件不同但物件還是有邏輯上的相等。在此由於 Patient 類別並未覆寫「equals」方法來實作邏輯相等的判斷程式，因此 p 所參考到的物件在邏輯上不會相等於 p2 所參考到的物件。除非將 Patient 類別改寫如下：
```
class Patient {

    String name;

    public Patient(String name) {
        this.name = name;
    }

    public boolean equals(Object obj) {
        if (this == obj) {
            return true;
        }
        if (obj instanceof Patient) {
            return ((Patient) obj).name.equals(this.name);
        }
        return false;
    }
}
```
選項 D，List 內原先就有 p2 所參考到的物件，因此可以使用「indexOf」方法來找到。

(170)
```
public class Test1 {

    static void doubling(Integer ref, int pv) {
        ref = 20;
        pv = 20;
    }

    public static void main(String[] args) {
        Integer iQbj = new Integer(10);
        int iVar = 10;
        doubling(iQbj++, iVar++);
        System.out.println(iQbj + "," + iVar);
    }
}
```

What is the result?

A(ans)
11, 11

B
10, 10

C
21, 11

D
20, 20

E
11, 12

題解
由於 Java 永遠為「pass by value」，因此 doubling 方法內對於參數 ref 和 pv 的變化並不會影響到外面的變數。這題只需注意到程式第 11 行，將 iQbj 和 iVar 做了加 1 的動作，因此輸出為：
11, 11

(171)
Customer.java
```
public class Customer {

    ElectricAccount acct = new ElectricAccount();

    public void useElectricity(double kWh) {
        acct.addKWh(kWh);
    }
}
```

ElectricAccount.java
```
public class ElectricAccount {

    private double kWh;
    private double rate = 0.07;
    private double bill;

    // line n1
}
```

How should you write methods in the ElectricAccount class at line n1 so that the member variable bill is always equal to the value of the member variable kwh multiplied by the member variable rate?

Any amount of electricity used by a customer (represented by an instance of the customer class) must contribute to the customer's bill (represented by the member variable bill) through the method useElectricity method. An instance of the customer class should never be able to tamper with or decrease the value of the member variable bill.

A

```
public void addKWh(double kWh){
    this.kWh += kWh;
    this.bill = this.kWh * this.rate;
}
```

B(ans)

```
public void addKWh(double kWh){
    if(kWh > 0){
        this.kWh += kWh;
        this.bill = this.kWh * this.rate;
    }
}
```

C

```
private void addKWh(double kWh){
    if(kWh > 0){
        this.kWh += kWh;
        this.bill = this.kWh * this.rate;
    }
}
```

D

```
public void addKWh(double kWh){
    if(kWh > 0){
        this.kWh += kWh;
        setBill(this.kWh);
    }
}

public void setBill(double kWh){
    bill = kWh * rate;
}
```

題解

選項 A，沒有判斷傳入的 kWh 參數的值是否大於 0，因此可能會有傳入負數的情形。

選項 B，正確答案。

選項 C，使用 private 來修飾 addKWh 方法，會導致 ElectricAccount 之外的其他類別無法存取。

選項 D，setBill 方法應該要改成 private 來修飾，避免帳單被竄改。

(172)

```
class Star {

    public void doStuff() {
        System.out.println("Twinkling Star");
    }
}

interface Universe {

    public void doStuff();
}

class Sun extends Star implements Universe {

    public void doStuff() {
```

```
        System.out.println("Shining Sun");
    }
}

public class Bob {

    public static void main(String[] args) {
        Sun obj2 = new Sun();
        Star obj3 = obj2;
        ((Sun) obj3).doStuff();
        ((Star) obj2).doStuff();
        ((Universe) obj2).doStuff();
    }
}
```

What is the result?

A(ans)
Shining Sun
Shining Sun
Shining Sun

B
Shining Sun
Twinkling Star
Shining Sun

C
Compilation fails

D
A ClassCastException is thrown at runtime
題解
程式第 23 行實體化出了一個 Sun 物件，接著將他向上轉型並呼叫其 doStuff 方法。由於 Sun 類別有覆寫(override)doStuff 方法，因此不論轉成哪個型態都會去呼叫到第 15 行的 doStuff 方法。

(172)

```
class MarksOutOfBoundsException extends
IndexOutOfBoundsException {
}

public class GradingProcess {

    void verify(int marks) throws IndexOutOfBoundsException {
        if (marks > 100) {
            throw new MarksOutOfBoundsException();
        }
        if (marks > 50) {
            System.out.print("Pass");
        } else {
            System.out.print("Fail");
        }
    }

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

```
        int marks = Integer.parseInt(args[2]);
        try {
            new GradingProcess().verify(marks);
        } catch (Exception e) {
            System.out.print(e.getClass());
        }
    }
}
```

And the command line invocation:

```
java GradingProcess 89 50 104
```

What is the result?

A
Pass

B
Fail

C(ans)
Class MarketOutOfBoundsException

D
Class IndexOutOfBoundsException

E
Class Exception

題解

此題輸入的指令會使得 main 方法的 args 參數為「{"89", "50", "104"}」。然後將索引位置為 2 的字串「104」轉成數值「104」並指派給 mark 變數儲存，再傳給 GradingProcess 物件的 verify 方法。由於 mark 的值大於 100，因此會拋出 MarksOutOfBoundsException。

(173)

SalesMan.java

```
package sales;

public class SalesMan {}
```

Product.java

```
package sales.products;


public class Product {}
```

Market.java

```
package Market;
// insert code here
public class Market {
    SalesMan sm;
    Product p;
}
```

Which code fragment, when inserted at line 2, enables the code to compile?

A
```
import sales.*;
```

B
```
import java.sales.products.*;
```

C
```
import sales;
import sales.products;
```

D
```
import sales.*;
import products.*;
```

E(ans)
```
import sales.*;
import sales.products.*;
```
題解
選項 A，「sales.*」包含 sales 套件底下的所有類別或是介面，但並不包含「sales.products」套件和其底下的類別與介面，因此 Product 類別還是沒有引入成功。
選項 B，套件(路徑)名稱錯誤。
選項 C，「import」是引入類別或是介面，無法直接引入套件。
選項 D，products 套件的(路徑)名稱錯誤。
選項 E，正確答案。

(174)
Which two are valid array declaration?

A(ans)
```
Object array[];
```

B
```
Boolean array[3];
```

C(ans)
```
int[] array;
```

D
```
Float[2] array;
```
題解
選項 A 是正確的陣列物件宣告方式。
選項 B 比較像是 C 語言宣告陣列的方式，但在 Java 語言中無法使用這樣的方式來宣告並指定陣列物件的長度。
選項 C 是正確的陣列物件宣告方式。
選項 D 理由同選項 B。

(174)
```
public static void main(String[] args) {
    Short s1 = 200;
    Integer s2 = 400;
```

```
    Long s3 = (long) s1 + s2; // line n1
    String s4 = (String) (s3 * s2); // line n2
    System.out.println("Sum is " + s4);
}
```

What is the result?

A
Sum is 600

B
Compilation fails at line n1.

C(ans)
Compilation fails at line n2.

D
A ClassCastException is thrown at line n1.

E
A ClassCastException is thrown at line n2.

題解

line n2 的「(s3 * s2)」無法強制轉型至 String 物件，原因在於數值和字串並沒有人任何的繼承關係，因此會發生編譯錯誤。
至於 Wrapper 類別與基本數值資料型別在正確的在型別的空間大小下，可以互相轉型，也能自動轉型(auto wrapping)。
若轉型的型態和要轉型的物件有繼承關係，在執行階段才會檢查物件實體的型態，如果無法轉型，會拋出 ClassCastException 例外。

(175)

Employee.java
```
public class Employee {

    public int salary;
}
```

Manager.java
```
public class Manager extends Employee {

    public int budget;
}
```

Director.java
```
public class Director extends Manager {

    public int stockOptions;
}
```

And given the follow main method:
```
public static void main(String[] args){
    Employee employee = new Employee();
    Manager manager = new Manager();
    Director director = new Director();
    // line n1
}
```

Which two options fail to compile when placed at line n1 of the main method?

A
```
employee.salary = 50_000;
```

B
```
director.salary = 80_000;
```

C(ans)
```
employee.budget = 200_000;
```

D
```
manager.budget = 1_000_000;
```

E(ans)
```
manager.stockOption = 500;
```

F
```
director.stockOptions = 1_000;
```

題解
選項 A，Employee 物件本身有 salary 欄位，因此不會有問題。
選項 B，Director 物件也是 Employee，有 salary 欄位，因此也不會有問題。
選項 C，Employee 物件並沒有 budget 欄位，編譯會有問題。
選項 D，Manager 物件本身有 budget 欄位，因此不會有問題。
選項 E，Manager 物件並沒有 stockOption 欄位，編譯會有問題。
選項 F，Director 物件本身有 stockOption 欄位，因此不會有問題。

(175)
```
class CD {

    int r;

    CD(int r) {
        this.r = r;

    }
}

class DVD extends CD {

    int c;

    DVD(int r, int c) {
        // line n1
    }
}
```

And given the code fragment:
```
DVD dvd = new DVD(10, 20);
```

Which code fragment should you use at line n1 to instantiate the dvd object successfully?

```
A
super.r = r;
this.c = c;

B
super(r);
this(c);
```

C(ans)

```
super(r);
this.c = c;
```

```
D
this.c = r;
super(c);
```

題解

由於第 15 行的 DVD 類別之建構子原先並沒有使用 super 或是 this 敘述來呼叫別的建構子，因此 Java 會在編譯的時候將它改成

```
DVD(int r, int c) {
    super();
}
```

因為 CD 類別並不存在「CD()」這樣的建構子，如此一來便會造成編譯錯誤，因此需要插入一些程式至 line n1 來解決這個問題。

選項 A 並沒有使用 super 或是 this 敘述來呼叫別的建構子，無法解決問題。

選項 B 用來呼叫自身其它建構子的 this 敘述會發生錯誤，因為不是在建構子的第一行呼叫。

選項 C 正確。

選項 D 用來呼叫自身父類別建構子的 super 敘述會發生錯誤，因為不是在建構子的第一行呼叫。

(176)

```
import java.util.ArrayList;
import java.util.List;

public class Whizlabs {

    public static void main(String[] args) {
        List<int> list = new ArrayList<>();
        list.add(21); list.add(13);
        list.add(30); list.add(11);
        list.removeIf(e -> e % 2 != 0);
        System.out.println(list);
    }
}
```

What is the output?

```
A
[21, 13, 11]

B
[30]

C
```

[]

D(ans)
Compilation fails due to error at line 7

E
Compilation tails due to error at line 10

題解

程式第 7 行使用到 Java 的泛型(generic)功能，但是用的並不正確，原因在於泛型不能用在基本資料型態。如果將這裡的「int」改為「Integer」的話，程式的執行結果為：[30]

程式第 10 行是 Java 8 的 Lambda 和 Collection 的新特性，removeIf 會把 Collection 物件內所有符合條件的元素刪除，在這裡就是把奇數全部刪除。

(177)

```
public class Test2 {

    public static void doChange(int... arr) {
        for (int pos = 0; pos < arr.length; pos++) {
            arr[pos] = arr[pos] + 1;
        }
    }

    public static void main(String[] args) {
        int[] arr = {10, 20, 30};
        doChange(arr);
        for (int x : arr) {
            System.out.print(x + ", ");
        }
        doChange(arr[0], arr[1], arr[2]);
        System.out.print(arr[0] + ", " + arr[1] + ", " +
arr[2]);
    }
}
```

What is the result?

A(ans)
11, 21, 31, 11, 21, 31

B
11, 21, 31, 12, 22, 32

C
12, 22, 32, 12, 22, 32

D
10, 20, 30, 10, 20, 30

題解

程式第 3 行使用了 Java 的 varargs 使得 doChange 的參數長度可以變動。

Java 傳遞變數內容的方式永遠為傳值(pass by value)。程式第 11 行直接將 arr 陣列物件變數所儲存的物件參考值傳給 doChange 方法使用，因此 doChange 可以對原先的 arr 陣列物件進行更動。程式第 15 行分別將 arr 整數陣列的整數元素傳給

doChange 方法，會把這些整數值複製另外一份空間出來給 doChange 的參數使用，因此數值在 doChange 的變化並不會影響到原先 arr 陣列物件的內容。

(178)

```
public class SumTest {

    public static void doSum(Integer X, Integer Y) {
        System.out.println("Integer sum is " + (X + Y));
    }

    public static void doSum(double x, double Y) {
        System.out.println("double sum is " + (x + Y));
    }

    public static void doSum(float x, float Y) {
        System.out.println("float sum is " + (x + Y));
    }

    public static void doSum(int x, int Y) {
        System.out.println("int sum is " + (x + Y));
    }

    public static void main(String[] args) {
        doSum(10, 20);
        doSum(10.0, 20.0);
    }
}
```

What is the result?

A
int sum is 30
float sum is 30.0

B(ans)
int sum is 30
double sum is 30

C
Integer sum is 30
double sum is 30.0

D
Integer sum is 30
float sum is 30.0

題解

doSumg 是一個多載(Overload)方法，Java 會自動根據傳入的參數數量和型態來決定要使用哪一個 doSumg 方法。
Java 的預設整數型態是 int，浮點數型態是 double，因此程式會執行第 15 行和第 7 行的 doSumg 方法。
若在程式碼內想使用 long 或是 float 的數值型態，可以直接在數值後接上「l」或是「f」來轉型。舉例來說，若將題目的程式第 20 和 21 行改成：

```
doSum(10l, 20l);
doSum(10.0f, 20.0f);
```

則程式執行結果會變成：

```
float sum is 30.0
float sum is 30.0
```
這裡長整數 long 型態的數值因為在 SumTest 類別中並沒有找到完全符合的 doSumg 方法，因此會自動往上轉型成有符合方法的 float 型態。
Integer 是 Java 的 Wrapper 類別，int 型態的數值可以被自動轉型(auto wrapping)成 Integer 物件。僅當傳入 int 型態的參數給 doSumg 方法時，若 doSumg 方法並沒有定義 int 和 long、float、double 等能夠讓 int 向上轉型的多載方法，才會使用 Integer 型態的 doSumg 多載方法。

(180)

Base.java
```
class Base {

    public void test() {
        System.out.println("Base ");
    }
}
```

DerivedA.java
```
class DerivedA extends Base {

    public void test() {
        System.out.println("DerivedA");
    }
}
```

DerivedB.java
```
class DerivedB extends Base {

    public void test() {
        System.out.println("DerivedB");
    }

    public static void main() {
        Base b1 = new DerivedB();
        Base b2 = new DerivedA();
        Base b3 = new DerivedB();
        b1 = (Base) b3;
        Base b4 = (DerivedA) b3;
        b1.test();
        b4.test();
    }
}
```

What is the result?

A
Base
DerivedA

B
Base
DerivedB

C(ans)

DerivedB
DerivedB

D
DerivedB
DerivedA

E
A classcast Except ion is thrown at runtime.

題解

「DerivedB.java」檔案的 main 方法中的 b1、b3 和 b4 這三個變數在第 12 行執行結束後，所參考到的物件實體都是從 DerivedB 類別實體化出來的。Base 類別的 test 方法在 DerivedB 類別中被覆寫，因此「DerivedB.java」檔案中的第 13~14 行程式會去執行第 3 行的 test 方法。

(181)

You are asked to develop a program for a shopping application, and you are given the following information:

1. The application must contain the classes Toy, EduToy, and consToy. The Toy class is the superclass of the other two classes.
2. The int caicuiatePrice (Toy t) method calculates the price of a toy.
3. The void printToy (Toy t) method prints the details of a toy.

Which definition of the Toy class adds a valid layer of abstraction to the class hierarchy?

A(ans)

```
public abstract class Toy {
    public abstract int calculatePrice(Toy t);
    public void printToy(Toy t) { /* code goes here */ }
}
```

B

```
public abstract class Toy {
    public int calculatePrice(Toy t);
    public void printToy(Toy t);
}
```

C

```
public abstract class Toy {
    public int calculatePrice(Toy t);
    public final void printToy(Toy t) { /* code goes here */ }
}
```

D

```
public abstract class Toy {
    public abstract int calculatePrice(Toy t) { /* code goes
here */ }
    public abstract void printToy(Toy t) { /* code goes here
*/ }
}
```

題解
選項 A，抽象方法沒有程式實作區塊，非抽象方法有程式實作區塊，因此是正確的。
選項 B，非抽象方法必須要有程式實作區塊。
選項 C，非抽象方法可以使用 final 修飾使其不可被覆寫(Override)，但其必須要有程式實作區塊。若是抽象方法，則不能使用 final 修飾，因為需要由繼承的類別來幫抽象方法實作程式區塊。
選項 D，抽象方法不能有程式實作區塊。

(182)
Which three statements describe the object-oriented features of the Java language?

A
Objects cannot be reused.

B(ans)
A subclass can inherit from a superclass.

C(ans)
Objects can share behaviors with other objects.

D
A package must contain more than one class.

E(ans)
Object is the root class of all other objects.

F
A main method must be declared in every class.
題解
選項 A，物件(object)是可以重複使用的。
選項 B，子類別(class)當然可以繼承父類別(不然怎麼會有父子關係)。
選項 C，物件可以和其它物件透過靜態(static)的類別方法、同樣的父類別方法或是介面(interface)方法來共享行為。
選項 D，並沒有規定套件(package)一定要至少有一個類別，什麼都不放，或是放介面也是可以。
選項 E，「Object」是所有類別物件的最上層。若類別沒有使用 extends 來指定繼承哪個類別的話，會自動繼承「Object」類別。
選項 F，並沒有規定每個類別一定要有「main」方法。

(183)
```
class X {
    public void printFileContent() {
        /* code goes here */
        throw new IOException();
    }
}
public class Test {
    public static void main(String[] args) {
        X xobj = new X();
        xobj.printFileContent();
    }
}
```

Which two modifications should you make so that
the code compiles successfully?

A(ans)
Replace line 8 with:
public static void main(String[] args) throws Exception {

B(ans)
Replace line 10 with:
```
try {
    xobj.printFileContent();
} catch (Exception e) {
} catch (IOException e) {
}
```

C(ans)
Replace line 2 with:
public void printFileContent() throws IOException {

D
Replace line 4 with:
```
throw IOException("Exception raised");
```

E
At line 11, insert
```
throw new IOException();
```
題解
第 5 行在 X 類別的「printFileContent」方法中拋出了新的「IOException」例外，IOException 是需要檢查的例外(checked exception)，因此必須要撰寫程式去處理它。
由於「printFileContent」方法中目前並沒有任何處理「IOException」的方式，因此可以利用選項 C，將「printFileContent」方法中所產生的「IOException」往外拋出。選項 D 缺少了「new」運算子來實體化出 IOException 物件，但就算有「new」運算子也無濟於事。
將「IOException」從「printFileContent」方法拋出後，其它有使用到「printFileContent」方法的地方就必須要去處理這個「IOException」，因此可以利用選項 A，繼續將「IOException」再從「main」方法往外拋出。選項 B 的 try-catch 用法是錯的，若把「Exception」和「IOException」調換，或是移除掉其中一組 catch，才會是正確的。
選項 E 雖然可以加至程式中，但無濟於事。

(184)
```
String[] strs = new String[2];
int idx = 0;
for (String s : strs) {
    strs[idx].concat(" element " + idx);
    idx++;
}
for (idx = 0; idx < strs.length; idx++) {
    System.out.println(strs[idx]);
}
```

What is the result?

A
Element 0
Element 1

B
Null element 0
Null element 1

C
Null
Null

D(ans)
A NullPointerException is thrown at runtime.
題解
程式第 6 行建立了長度為 2 的字串陣列，陣列的每個元素會自動被初始化為「null」。
程式第 9 行使用到字串物件的「concat」方法，但由於變數所存的參考為 null，因此會發生 NullPointerException。

(185)
Which of the following data types will allow the following code snippet to compile?

```
float i = 4;
float j = 2;
_____ z = i + j;
```

Which of the following data types will allow the following code snippet to compile?

A
Long

B(ans)
Double

C
Int

D(ans)
Float

E
Byte
題解
單精準浮點數(float)為 32 位元的浮點數，雙精準浮點數(double)為 64 位元的浮點數。由於 i 變數和 j 變數都是 32 位元的單精準浮點數，因此它們的運算結果也為 32 位元的單精準浮點數，也可以使用 64 位元的雙精準浮點數型態來儲存。所以 z 變數可以使用的資料型態為 float 和 double。

(186)

```
public class Whizlabs {
    private String name;
    private boolean pass;
    public static void main(String[] args) {
        Whizlabs wb = new Whizlabs();
```

```
        System.out.print("name = " + wb.name);
        System.out.print(",pass = " + wb.pass);
    }
}
```

What would be the output, if it is executed as a program?

A
name =, pass =

B
name = null, pass = null

C(ans)
name = null, pass = false

D
name = null pass = true

E
Compile error.

題解

Java 的類別或是物件欄位會按照不同資料型態自動給定初始值。初始值如下：

String、Object：null
byte、short、int、long、float、double：0
char：\0
boolean：false

(187)

* AssertionError
* ArithmeticException
* ArrayIndexOutofBoundsException
* FileNotFoundException
* IllegalArgumentException
* IOError
* IOException
* NumberFormatException
* SQLException

Which option lists only those classes that belong to the unchecked exception category?

A(ans)
AssertionError, ArrayIndexOutOfBoundsException, ArithmeticException

B
AssertionError, IOError, IOException

C
ArithmeticException, FileNotFoundException, NumberFormatException

D
FileNotFoundException, IOException, SQLException

E
ArrayIndexOutOfBoundException, IllegalArgumentException, FileNotFoundException

題解

Java 將例外狀況分成兩種類型，一為需要檢查的例外(checked exception)，另一為不需檢查的例外(unchecked exception)。

需要檢查和不需檢查的例外差別在於撰寫程式的時候，需要檢查的例外必須使用「try-cache」結構或是「throws」關鍵字來處理例外，否則程式會無法編譯成功，而不需檢查的例外則不管有沒有寫程式去處理都可以編譯成功。

不需檢查的例外包含 RuntimeException、Error 以及所有其它繼承它們的例外(或是錯誤)，常見的有 ArrayIndexOutOfBoundsException 和 NullPointerException。其餘的例外都是需要檢查的例外，常見的有 IOException 和 SQLException。

選項 A 全都是不需檢查的例外。

選項 B 的 IOException 是需要檢查的例外。

選項 C 的 FileNotFoundException 是需要檢查的例外。

選項 D 全都是需要檢查的例外。

選項 E 的 FileNotFoundException 是需要檢查的例外。

(188)

```
String[] arr = {"A", "B", "C", "D"};
for (int i = 0; i < arr.length; i++) {
    System.out.print(arr[i] + " ");
    if (arr[i].equals("C")) {
        continue;
    }
    System.out.println("Work down");
    break;
}
```

What is the result?

A
A B C Work done

B
A B C D Work done

C(ans)
A Work done

D
Compilation fails

題解

第一次執行第 7 行的 for 迴圈，i 為 0，arr[i]為 A，所以第 9 行的 if 條件式不成立，程式執行到第 13 行的「break;」就直接跳出迴圈。所以只輸出了「A Work done」。

(189)

```
public class MyClass{
    public static void main(String[] args){
        while(int ii = 0; ii < 2){
            ii++;
```

```
            System.out.println("ii = " + ii);
        }
    }
}
```

What is the result?

A
ii = 1
ii = 2

B(ans)
Compilation fails

C
The program prints nothing

D
The program goes into an infinite loop with no output

E
The program goes to an infinite loop outputting:
ii = 1
ii = 1

題解
此題的 while 迴圈用法錯誤，會造成編譯錯誤。如果要修正，可以改寫成：

```
public class MyClass{
    public static void main(String[] args){
        int ii = 0;
        while(ii < 2){
            ii++;
            System.out.println("ii = " + ii);
        }
    }
}
```

或是使用 for 迴圈改寫成：

```
public class MyClass{
    public static void main(String[] args){
        for(int ii = 0; ii < 2;){
            ii++;
            System.out.println("ii = " + ii);
        }
    }
}
```

(190)

```
class Test {
    int sum = 0;
    public void doCheck(int number) {
        if (number % 2 == 0) {
            break;
        } else {
            for (int i = 0; i < number; i++) {
                sum += i;
            }
        }
```

```
    }

    public static void main(String[] args) {
        Test obj = new Test();
        System.out.println("Red " + obj.sum);
        obj.doCheck(2);
        System.out.println("Orange " + obj.sum);
        obj.doCheck(3);
        System.out.println("Green " + obj.sum);
    }
}
```

What is the result?

A
Red 0
Orange 0
Green 3

B
Red 0
Orange 0
Green 6

C
Red 0
Orange 1

D
Green 4

E(ans)
Compilation fails
題解
「break;」敘述只能用在迴圈和「switch」結構中，因此這題會編譯錯誤。

(191)
```
interface CanFly {
    String type = "A";
    void fly();
    -------  String getType() {
        return type;
    }
}
```

A
abstract

B
public

C(ans)
Default

D

It will not compile with any as interfaces cannot have non abstract methods.

E
It will compile without filling the blank.

題解

這題是在考 Java 8 的新特性，也就是介面(interface)的預設(default)方法和靜態(static)方法。

若要在介面實作預設方法，需使用「default」修飾字來定義方法，預設方法只能由實作這個介面的類別所實體化出的物件來使用；若要在介面實作靜態方法，需使用「static」修飾字來定義方法，靜態方法只能透過介面來使用，無法透過實作這個介面的物件實體來使用。

因此在這個題目中，可以填入選項 C 的「default」，使「getType()」變成預設方法。

(192)

```
String[][] arra = new String[3][];
arra[0] = new String[]{"rose", "lily"};
arra[1] = new String[]{"apple", "berry", "cherry", "grapes"};
arra[2] = new String[]{"beans", "carrot", "potato"};
// insert code fragment here
```

Which code fragment when inserted at line '// insert code fragment here', enables the code to successfully change arra elements to uppercase?

A(ans)

```
for(int i = 0; i < arra.length; i++){
    for(int j = 0; j < arra[i].length; j++){
        arra[i][j] = arra[i][j].toUpperCase();
    }
}
```

B

```
for(int i = 0;i < 3; i++){
    for(int j=0; j < 4; j++){
        arra[i][j] = arra[i][j].toUpperCase();
    }
}
```

C

```
for(String a[] : arra[][]){
    for(String x : a[]){
        x.toUpperCase();
    }
}
```

D

```
for(int i : arra.length){
    for(String x : arra){
        arra[i].toUpperCase();
    }
}
```

題解

選項 A，索引值範圍和修改陣列的方式都是正確的。

選項 B，內迴圈錯誤的固定索引範圍會使程式拋出 ArrayIndexOutOfBoundsException。
選項 C，「toUpperCase」方法並不會直接修改字串的內容，而是會將修改後的字串回傳，所以這樣做是沒有效果的。
選項 D 有嚴重錯誤的語法，for 迴圈無法這樣使用。

(193)
You are developing a banking module. You have developed a class named ccMask that has a maskcc method

```
class CCMask {
    public static String maskCC(String creditCard) {
        String x = "XXXX-XXXX-XXXX-";
        //line n1
    }
    public static void main(String[] args) {
        System.out.println(maskCC("1234-5678-9101-1121"));
    }
}.
```

You must ensure that the maskcc method returns a string that hides all digits of the credit card number except the four last digits (and the hyphens that separate each group of four digits).

Which two code fragments should you use at line n1, independently, to achieve this requirement?

A
```
StringBuilder sb = new StringBuilder(creditCard);
sb.substring(15, 19);
return x + sb;
```

B(ans)
```
return x + creditCard.substring(15, 19);
```

C(ans)
```
StringBuilder sb = new StringBuilder(x);
sb.append(creditCard, 15, 19);
return sb.toString();
```

D
```
StringBuilder sb = new StringBuilder(creditCard);
StringBuilder s = sb.insert(0, x);
return s.toString();
```

題解
選項 A 使用到 StringBuilder 物件的「substring」方法，看似可以，但「substring」方法是直接回傳新的子字串，實際上並不會改變 StringBuilder 物件本身的內容。因此這樣做沒有辦法輸出我們想要的結果。
選項 B 解決了選項 A 的問題。
選項 C 使用了 StringBuilder 物件來串接出我們想要的結果。
選項 D 是直接在原本的 creditCard1 變數所儲存的字串前直接插入「XXXX-XXXX-XXXX-」，這樣做不是我們想要的結果。

(194)

```
public class Whizlabs{
    public static void main(String[] args){
        int sum = 0;
                for (int x = 0; x <= 10; x++)
            sum += x;
        System.out.print("Sum for 0 to" + x);
        System.out.print(" = " + sum);
    }
}
```

Which is true?

A
Sum for 0 to 0 = 55

B
Sum for 0 to 10 = 55

C
Compilation fails due to error on line 6.

D(ans)
Compilation fails due to error on line 7.

E
An Exception is thrown at the runtime.

題解
程式第 7 行使用到第 5 行 for 迴圈內宣告的 x 變數，由於 x 變數只在 for 迴圈內才可以被有效地存取，而程式第 7 行已經在 for 迴圈之外，因此會發生編譯錯誤。

(194)

```
class A {
}
class B {
}
interface X {
}
interface Y {
}
```

Which two definitions of class C are valid?

A(ans)
class C extends A implements X { }

B
class C implements Y extends B { }

C
class C extends A, B { }

D
class C implements X, Y extends B { }

E(ans)

```
class C extends B implements X, Y { }
```

題解

類別(class)可以使用「implements」實作多個介面(interface)，但只可以使用「extends」繼承單一個類別，且「extends」必須在「implements」之前使用。

(195)

MyString.java

```
package p1;
class MyString {
    String msg;
    MyString(String msg) {
        this.msg = msg;
    }
}
```

Test.java

```
package p1;

public class Test {

    public static void main(String[] args) {
        System.out.println("Hello " + new StringBuilder("Java
SE 8"));
        System.out.println("Hello " + new MyString("Java SE
8"));
    }
}
```

What is the result?

A

```
Hello Java SE 8
Hello Java SE 8
```

B

```
Hello java.lang.StringBuilder@<<hashcode1>>
Hello p1.MyString@<<hashcode2>>
```

C(ans)

Hello Java SE 8
Hello p1.MyString@<<hashcode>>

D

Compilation fails at the Test class.

題解

運算子「+」若其中一個運算元為字串，那麼它將會做字串連接，否則為加法運算。任何的物件，都擁有「toString()」方法，可以轉成字串。預設的「toString()」方法於 Object 類別中，程式實作如下：

```
public String toString() {
    return getClass().getName() + "@" +
Integer.toHexString(hashCode());
}
```

由於 MyString 類別並未覆寫 Object 的「toString()」方法，因此會直接使用原本的程式實作。

所以這個題目的實際輸出為：

```
Hello Java SE 8
Hello p1.MyString@15db9742
```
「15db9742」為雜湊值，在不同環境下可能有不一樣的結果。

(196)
```
System.out.println("5 + 2 = " + 3 + 4);
System.out.println("5 + 2 = " + (3 + 4));

What is the result?

A
5 + 2 = 34
5 + 2 = 34

B
5 + 2 + 3 + 4
5 + 2 = 7

C
7 = 7
7 + 7
```

D(ans)
```
5 + 2 = 34
5 + 2 = 7
```
題解
運算子「+」若其中一個運算元為字串，那麼它將會做字串連接，否則為加法運算。
由於優先順序的關係，題目第 1 行會等同：
```
System.out.println(("5 + 2 = " + 3) + 4);
```
因此「3」和「4」會被當作字串進行連接。
若題目是寫成：
```
System.out.println(5 + 2 + " = 3 + 4");
```
等同：
```
System.out.println((5 + 2) + " = 3 + 4");
```
因此會輸出：
```
7 = 3 + 4
```

(197)
```
public class Test {

    public static void main(String[] args) {
        int numbers[];
        numbers = new int[2];
        numbers[0] = 10;
        numbers[1] = 20;

        numbers = new int[4];
        numbers[2] = 30;
        numbers[3] = 40;
        for(int x : numbers){
            System.out.print(" " + x);
        }
    }
}
```

What is the result?

A
10 20 30 40

B(ans)
0 0 30 40

C
Compilation fails

D
An exception is thrown at runtime

題解
程式第 9 行，將新長度為 4 的整數陣列指派給 numbers 變數，因此在這之前長度為的 2 的整數陣列沒有被任何的變數所指向(此陣列空間可稱為孤島)，其所佔用的記憶體會自動在稍候被 Java 的垃圾回收(Garbage Collection)機制給回收重新使用。
數值陣列的裡的數值，一開始的初始值為 0。程式只有在第 10~11 行時更改了 numbers 陣列物件索引 2 和 3 的值。因此最後輸出為「 0 0 30 40」。

(198)
```
abstract class X {

    public abstract void methodX();
}

interface Y {

    public void methodY();
}
```

Which two code fragments are valid?

A
```
class Z extends X implements Y{
    public void methodZ(){}
}
```

B(ans)
```
abstract class Z extends X implements Y{
    public void methodZ(){}
}
```

C
```
class Z extends X implements Y{
    public void methodX(){}
}
```

D(ans)
```
abstract class Z extends X implements Y{
}
```

E
```
class Z extends X implements Y{
    public void methodY(){}
```

```
}
```

題解

X 是抽象類別，Y 是介面，要成功建立 Z 這個類別去繼承 X 與實作 Y。如果 Z 並不是抽象類別的話，必須實作出 X 的抽象方法以及 Y 定義的方法介面，因此選項 A、C、E 都不行。如果 Z 是抽象類別，就能允許沒有實作出程式區塊的方法。

(199)

```
class Product {

    double price;
}

public class Test {

    public void updatePrice(Product product, double price) {
        price = price * 2;
        product.price = product.price + price;
    }

    public static void main(String[] args) {

        Product prt = new Product();
        prt.price = 200;
        double newPrice = 100;

        Test t = new Test();
        t.updatePrice(prt, newPrice);
        System.out.println(prt.price + " : " + newPrice);
    }
}

What is the result?

A
200.0 : 100.0

B
400.0 : 200.0

C(ans)
400.0 : 100.0

D
Compilation fails.
```

題解

這題是在考 Java 永遠為「pass by value」的概念。

第 20 行同時將基本型態的區域變數「newPrice」和物件參考的變數「prt」當作參數傳給「updatePrice」方法。此時「newPrice」和「prt」的值都會被複製一份出來，「prt」的值為 Product 物件的參考(類似記憶體位址)。

「updatePrice」方法會將「newPrice」的值乘 2，再加給「prt」物件的「price」變數。

第 21 行將「prt」物件的「price」變數的值和「newPrice」變數的值輸出。「prt」物件的「price」變數的值會是執行「updatePrice」方法時改變的結果，而「main」方法的「newPrice」變數並不會被改變。

(200)

```
public static void main(String[] args) {
    ArrayList myList = new ArrayList();
    String[] myArray;
    try {
        while (true) {
            myList.add("My String");
        }
    } catch (RuntimeException re) {
        System.out.println("Caught a RuntimeException");
    } catch (Exception e) {
        System.out.println("Caught a Exception");
    }
    System.out.println("Ready to use");
}
```

What is the result?

A
Execution terminates in the first catch statement, and caught a RuntimeException is printed to the console.

B
Execution terminates In the second catch statement, and caught an Exception is printed

C(ans)
A runtime error is thrown in the thread "main".

D
Execution completes normally, and Ready to use is printed to the console.

E
The code fails to compile because a throws keyword is required.

題解

第 11 行的 while 迴圈會不斷執行，因為沒有任何的中止條件，所以會不斷加入「My String」字串至 myList 這個 ArrayList 物件中。當記憶體不夠將字串添加至 ArrayList 物件的時候，就會產生 OutOfMemoryError。由於 try-cache 並沒有處理 Error 類別底下的 OutOfMemoryError，因此會繼續將 OutOfMemoryError 往外拋出，但是並沒有任何程式去處理這個 OutOfMemoryError，執行緒變會原封不動地將其拋出，然後中止執行。

(201)

```
class Vehicle {

    int x;

    Vehicle() {
        this(10); // line n1
    }

    Vehicle(int x) {
        this.x = x;
```

```
    }
}

class Car extends Vehicle {

    int y;

    Car() {
        super();
        this(20); // line n2
    }

    Car(int y) {
        this.y = y;
    }

    public String toString() {
        return super.x + ":" + this.y;
    }
}
```

And given the code fragment:

```
Vehicle y = new Car();
System.out.println(y);
```

What is the result?

A
10:20

B
0:20

C
Compilation fails at line n1

D(ans)
Compilation fails at line n2
題解
第 6 行(line n1)用「this」敘述呼叫了 Vehicle 物件的另一個建構子，而 Vehicle 物件的確擁有「Vehicle(int x)」建構子，因此這是沒問題的。
第 20 行(line n2)也用「this」敘述呼叫了 Vehicle 物件的另一個建構子，但並非在建構子的第 1 行使用「this」敘述，而會造成編譯錯誤。

(202)

```
int[] intArr = {15, 30, 45, 60, 75};
intArr[2] = intArr[4];
intArr[4] = 90;
```

What are the values of each element in intArr after this code has executed?

A
15, 60, 45, 90, 75

B
15, 90, 45, 90, 75

C(ans)
15, 30, 75, 60, 90

D
15, 30, 90, 60, 90

E
15, 4, 45, 60, 90

題解
這題要注意 Java 的陣列索引是從 0 開始的。
第 2 行程式執行後，陣列的內容為{15, 30, 75, 60, 75}。
第 3 行程式執行後，陣列的內容為{15, 30, 75, 60, 90}。

(203)
```
public class Employee {

    String name;
    boolean contract;
    double salary;

    Employee() {
        // line n1
    }

    public String toString() {
        return name + ":" + contract + ":" + salary;
    }

    public static void main(String[] args) {
        Employee e = new Employee();
        // line n2
        System.out.print(e);
    }
}
```

Which two modifications, when made independently, enable the code to print joe:true:100.0?

A(ans)
Replace line n2 with:
```
e.name = "Joe";
e.contract = true;
e.salary = 100;
```

B
Replace line n2 with:
```
this.name = "Joe";
this.contract = true;
this.salary = 100;
```

C(ans)
Replace line n1 with:

```
this.name = new String("Joe");
this.contract = new Boolean(true);
this.salary = new Double(100);
```

D
Replace line n1 with:
```
name = "Joe";
contract = TRUE;
salary = 100.0f;
```

E
Replace line n1 with:
```
this("Joe", true, 100);
```
題解
由於 name、contract 和 salary 都是 Employee 物件的物件變數，需要先實體化之後透過物件參考才可以存取，因此選項 A 是可以的。
選項 B 由於 main 方法是屬於類別(靜態)方法，並沒有實體，無法使用 this 關鍵字。
選項 C 是在 Employee 物件的建構子內直接修改其物件變數的值，因此是正確的。
選項 D 並沒有「TRUE」這個 boolean 值。
選項 E 並沒有實作「this(String name, boolean contract, double 100)」這樣的建構子。

(204)
```
public abstract class Shape {

    private int x;
    private int y;

    public abstract void draw();

    public void setAnchor(int x, int y) {
        this.x = x;
        this.x = y;
    }
}
```

Which two classes use the shape class correctly?

A
```
public class Circle implements Shape{
    private int radius;
}
```

B(ans)
```
public abstract class Circle extends Shape{
    private int radius;
}
```

C
```
public class Circle extends Shape{
    private int radius;
    public void draw();
}
```

D

```
public abstract class Circle implements Shape{
    private int radius;
    public void draw();
}
```

E(ans)

```
public class Circle extends Shape{
    private int radius;
    public void draw() { /* code here */ }
}
```

F

```
public abstract class Circle implements Shape{
    private int radius;
    public void draw() { /* code here */ }
}
```

題解

這題是在考抽象類別(abstract class)的用法。

抽象類別內允許沒有實作的抽象方法。由於抽象類別也是類別，因此無法使用實作介面(interface)的「implements」關鍵字來實作，所以選項 A、D、F 都是錯誤的。另外，抽象類別的抽象方法必須被繼承該抽象類別的非抽象類別來實作，因此選項 C 也是錯誤的。

(205)

```
public class Vowel {

    private char var;

    public static void main(String[] args) {
        char var1 = 'a';
        char var2 = var1;
        var2 = 'e';

        Vowel obj1 = new Vowel();
        Vowel obj2 = obj1;
        obj1.var = 'i';
        obj2.var = 'o';
        System.out.println(var1 + ", " + var2);
        System.out.println(obj1.var + ", " + obj2.var);
    }
}
```

A

```
a, e
i, o
```

B(ans)

```
a, e
o, o
```

C

```
e, e
I, o
```

D

e, e
o, o

題解

這裡是考 Java 永遠為「pass by value」的觀念。
var1 和 var2 是基本型別變數，直接儲存資料的數值；而 var 是物件參考變數，儲存物件的參考(Reference)，類似記憶體位址的值。在傳遞變數的過程中，Java 只會傳遞變數的「值」。
第 6~8 行，var1 一開始儲存'a'，然後 var1 再將'a'傳給 var2 儲存，最後再讓 var2 儲存'e'這個值。所以此時 var1 存的是'a'，var2 存的是'e'。
第 10~11 行，obj1 指到一個新實體化的 Vowel 物件，接著再將這個物件的參考也讓 obj2 儲存。所以此時 obj1 和 obj2 這兩個變數所指到的物件是同一個。
第 12~13 行，只是在修改同一個 Vowel 物件的 var 變數，最後 Vowel 物件的 var 變數存的值為'o'。

(206)

```
public class Natural {

    private int i;

    void disp() {
        while (i <= 5) {
            for (int i = 1; i <= 5;) {
                System.out.print(i + " ");
                i++;
            }
            i++;
        }
    }

    public static void main(String[] args) {
        new Natural().disp();
    }
}
```

What is the result?

A
Prints 1 2 3 4 5 once

B
Prints 1 3 5 once

C
Prints 1 2 3 4 5 five times

D(ans)
Prints 1 2 3 4 5 six times

E
Compilation fails

題解

第 3 行宣告了物件變數 i，第 7 行的 for 迴圈又宣告了一個變數 i，因此 for 的變數 i 會遮蔽掉物件的變數 i。無論 for 怎麼對它的變數 i 做任何更動，都不會影響到物件的變數 i。

物件變數若沒有給定初始值，則預設為 0。第 6 行開始的 while 迴圈與第 11 行的「i++;」決定了 while 迴圈的執行次數，i 的變化為 0,1,2,...,5,6，只有在小於等於 5 的時候才會執行 while 迴圈的內容，因此第 7 行的 for 迴圈被執行了 6 次。每執行一次 for 迴圈，會印出「1 2 3 4 5 」。

(207)

```
public static void main(String[] args) {
    int iArray[] = {65, 68, 69};
    iArray[2] = iArray[0];
    iArray[0] = iArray[1];
    iArray[1] = iArray[2];
    for (int element : iArray) {
        System.out.print(element + " ");
    }
}
```

A
68, 65, 69

B(ans)
68, 65, 65

C
65, 68, 65

D
65, 68, 69

E
Compilation fails

題解
程式第 3 行執行後，陣列的值為{65, 68, 65}。
程式第 4 行執行後，陣列的值為{68, 68, 65}。
程式第 5 行執行後，陣列的值為{68, 65, 65}。

(208)

```
public class Test3 {

    public static void main(String[] args) {
        String names[] = new String[3];
        names[0] = "Mary Brown";
        names[1] = "Nancy Red";
        names[2] = "Jessy Orange";
        try {
            for (String n : names) {
                try {
                    String pwd = n.substring(0, 3) + n.substring(6, 10);
                    System.out.println(pwd);
                } catch (StringIndexOutOfBoundsException sie) {
                    System.out.println("String out of limits");
                }
            }
        } catch (ArrayIndexOutOfBoundsException e) {
            System.out.println("Array out of limits");
        }
```

```
    }
}
```

What is the result?

A(ans)
Marrown
String out of limits
JesOran

B
Marrown
String out of limits
Array out of limits

C
Marrown
String out of limits

D
Marrown
NanRed
JesOran

題解

names[0]的長度是 10，索引範圍是 0~9；names[1]的長度是 9，索引範圍是 0~8；names[2]的長度是 12，索引範圍是 0~11。當字串索引範圍不正確的時候會拋出「StringIndexOutOfBoundsException」例外。

程式第 11 行有用到字串物件的「substring」方法來取的子字串，取得 names 字串陣列中每個字串的子字串，子字串的字元索引範圍是 0~2 和 6~9，所以長度未滿 10 的 names[1]將會拋出「StringIndexOutOfBoundsException」例外，而其他的字串則可以成功地進行子字串的串接。

(209)

```
int num[][] = new int[1][3];
for (int i = 0; i < num.length; i++) {
    for (int j = 0; j < num[i].length; j++) {
        num[i][j] = 10;
    }
}
```

Which option represents the state of the num array after successful completion of the outer loop?

A(ans)
num[0][0]=10
num[0][1]=10
num[0][2]=10

B
num[0][0]=10
num[1][0]=10
num[2][0]=10

C

```
num[0][0]=10
num[0][1]=0
num[0][2]=0
```

D
```
num[0][0]=10
num[0][1]=10
num[0][2]=10
num[0][3]=10
num[1][0]=10
num[1][1]=10
num[1][2]=10
num[1][3]=10
```

題解

第 1 行宣告並實體化了一個長度為 1x3 的「num」整數陣列，所以一共包含了 3 個整數空間。這裡要注意 Java 的陣列是從索引 0 開始算起，所以 num 陣列變數的有效存取範圍是 num[0][0~2]。接下來的迴圈將陣列裡的每個整數空間都存了 10 進去，因此答案為選項 A。

(210)
```
int x = 10;
if (x > 10) {
    System.out.println(">");
} else if (x < 10) {
    System.out.println("<");
} else {
    System.out.println("=");
}
```

Which of the following is equivalent to the above code fragment?

A
```
System.out.printLn(x > 10 ? ">" : "<" : "=");
```

B
```
System.out.println(x > 10 ? ">" ? "<" : "=");
```

C(ans)
```
System.out.println(x > 10 ? ">" : x < 10 ? "<" : "=");
```

D
```
System.out.printLn(x > 10 ? ">" ? "<" ? "=");
```

E
None of the above

題解

這題是在測驗三元運算子(?:)的用法。三元運算子在某些情況下可以取代 if-else 結構，而且可以獲得比 if-else 還要更好的效能。三元運算子的格式如下：

判斷式 ? 判斷式成立時要回傳的值 : 判斷式不成立時要回傳的值

三元運算子的「?」和「:」一定是成對出現的，因此選項 A、B、D 都錯誤，而選項 C 的條件邏輯符合題目給的條件程式。

(211)

```
public class CharToStr {
    public static void main(String[] args) {
        String str1 = "Java";
        char str2[] = {'J', 'a', 'v', 'a'};
        String str3 = null;
            for (char c : str2) {
                str3 = str3 + c;
            }
            if (str1.equals(str3))
                System.out.print("Successful");
            else
                System.out.print("Unsuccessful");
    }
}
```

What is result?

A
Successful

B(ans)
Unsuccessful

C
Compilation fails

D
An exception is thrown at runtime

題解

第 5 行的 str3 一開始為 null，接著在第 7 行直接使用「+」運算子與字元做字串連接，這樣的方式在 Java 是可被允許的。參考至 null 的字串在進行字串連接時會自動被改為長度為 4 的「null」字串，因此這題的 str3 變數串接到最後的結果為「nullJava」，「Java」和「nullJava」明顯不是一樣的字串，因此輸出為「Unsuccessful」。

以下再給一個例子：

```
String str = null;
System.out.println(str + "c");
System.out.println(null + "c");
System.out.println(str + 'c');
System.out.println('n' + 'u' + 'l' + 'l' + 'c');
System.out.println((String)null + 'c');
//System.out.println(null + 'c');
```

輸出結果為：

```
nullc
nullc
nullc
542
nullc
```

這裡要注意的地方是，「+」運算子的運算元必須要至少有一個是字串或是物件，才會被當作是字串連接，否則為加法運算。不知型態的 null 與字元 c(字元值為 99)無法做加法運算，因此會發生編譯錯誤。

(212)

```
int[] array = {1, 2, 3, 4, 5};
```

```
for (int i : array) {
    if (i < 2) {
        keyword1;
    }
    System.out.println(i);
    if (i == 3) {
        keyword2;
    }
}
```

What should keyword1 and keyword2 be respectively, in oreder to produce output 2345?

A
continue, break

B
break, break

C
break, continue

D(ans)
continue, continue
題解
continue 的作用是直接執行下一次的迴圈；break 的作用是直接跳出迴圈。array 的數值為 1~5，但是 1 沒有輸出，也就是說在 i 數值小於 2 的時候，迴圈就要直接執行下一次的迴圈，因此 keyword1 為 continue。輸出了「23」之後還有「45」，可知迴圈在 i 等於 3 的最後，迴圈並未跳出，還能繼續執行到最後，因此 keyword2 也為 continue。

(213)
```
class Alpha {
    public String doStuff(String msg) {
        return msg;
    }
}

class Beta extends Alpha {
    public String doStuff(String msg) {
        return msg.replace('a', 'e');
    }
}

class Gamma extends Beta {
    public String doStuff(String msg) {
        return msg.substring(2);
    }
}
```

And the code fragment of the main() method,
```
List<Alpha> strs = new ArrayList<Alpha>();
strs.add(new Alpha());
strs.add(new Beta());
strs.add(new Gamma());
```

```
for (Alpha t : strs) {
    System.out.println(t.doStuff("Java"));
}
```

What is the result?

A
Java
Java
Java

B(ans)
Java
Jeve
va

C
Java
Jeve
ve

D
Compilation fails

題解

由於「doStuff」是物件方法，因此會發生覆寫(Override)的作用。不管物件如何向上轉型，都需從它的實體物件的所屬類別型態開始向上尋找最後發生覆寫的方法。在這個題目中，「Alpha」、「Beta」、「Gamma」都有覆寫「doStuff」物件方法，所以它們的實體物件都會使用到自己的「doStuff」物件方法。因此輸出結果為：

```
Java
Jeve
va
```

(214)
Which two items can legally be contained within a java class declaration?

A
An import statement

B(ans)
A field declaration

C
A package declaration

D(ans)
A method declaration

題解

不考慮註解的話，選項 C 的「package」必須在「.java」檔案中最上方，選項 A 的「import」位置應緊接在「.java」檔案中的「package」之下，若沒有「package」，則會在「.java」檔案中的最上方。程式範例如下：

```
package org.magiclen;

import java.util.Scanner;
```

```
import java.util.Base64;

public class Test {
    ...
}

import java.util.Scanner;
import java.util.Base64;

public class Test {
    ...
}
```

選項 B 所指的欄位(field)，代表所有的變數和常數，存取時在「.」之後不需加括號。
選項 D 指的方法(method)，存取時在「.」之後需要加括號。程式範例如下：

```
public class Plane {

    private final int width = 10, height = 8; // 欄位

    private static int computePlaneArea() { // 方法
        Plane plane = new Plane();
        return plane.width * plane.height;
    }

    public static void main(String[] args) { // 方法
        System.out.println(computePlaneArea());
    }
}
```

(215)

```
public class Test {

    static int count = 0;
    int i = 0;

    public void changeCount() {
        while (i < 5) {
            ++i;
            count++;
        }
    }

    public static void main(String[] args) {
        Test check1 = new Test();
        Test check2 = new Test();
        check1.changeCount();
        check2.changeCount();
        System.out.println(check1.count + " : " + check2.count);
    }
}
```

What is the result?

A(ans)

10 : 10

B
5 : 5

C
5 : 10

D
Compilation fails
題解
「count」是類別(靜態)變數，就算 Test 類別被實體化成許多個物件，用來儲存「count」的記憶體空間也都還是一開始的那個。
check1 和 check2 物件分別執行一次「changeCount」方法，將 Test 類別的「count」類別變數從 0 開始，加 1 加了 10 次。因此最後「count」類別變數儲存的值為 10，輸出「10 : 10」。

(216)
```
public class Calculator {
    public static void main(String[] args) {
        int num = 5;
        int sum;

        do {
            sum += num;
        } while ((num--) > 1);
        System.out.println("The sum is " + sum + ".");
    }
}
```

What is the result?

A
The sum is 2

B
The sum is 14

C
The sum is 15

D
The loop executes infinite times

E(ans)
Compilation fails
題解
第 4 行宣告了一個 sum 整數變數，但沒有初始化。第 7 行的「sum += num;」可以拆解成：
sum = sum + num;
由於 sum 變數並未初始化就要被取值作加法運算，因此會發生編譯錯誤。

(217)
```
class Sports {

    int num_players;
```

```
    String name, ground_condition;

    Sports(int np, String sname, String sground) {
        num_players = np;
        name = sname;
        ground_condition = sground;
    }
}

class Cricket extends Sports {

    int num_umpires;
    int num_substitutes;

    //insert code here
}
```

Which code fragment can be inserted at line "//insert code here" to enable the code to compile?

A(ans)

```
Cricket() {
    super(11, "Cricket", "Condidtion OK");
    num_umpires = 3;
    num_substitutes= 2;
}
```

B

```
Cricket() {
    super.ground_condition = "Condition OK";
    super.name= "Cricket";
    super.num_players = 11;
    num_umpires = 3;
    num_substitutes= 2;
}
```

C

```
Cricket() {
    this(3,2);
    super(11, "Cricket", "Condidtion OK");
}
Cricket(int nu, int ns) {
    this.num_umpires = nu;
    this.num_substitutes= ns;
}
```

D

```
Cricket() {
    this.num_umpires = 3;
    this.num_substitutes = 2;
    super(11, "Cricket", "Condidtion OK");
}
```

題解

這題是考建構子的觀念。每個類別都至少擁有一個建構子，若類別並沒有實作建構子，則其在編譯時還是會預設加入一段建構子程式。以題目「Cricket」這個類別為例，一開始並沒有實作建構子，所以編譯器會自動加入以下這段程式：

```
public Cricket(){}
```
再來，所有的建構子在執行的時候都會最先呼叫父類別的建構子或是自己類別其他的建構子。如果建構子內並沒有指定要先呼叫哪個建構子，那會編譯器會一律在建構子的第一行程式加上「super();」敘述。以題目「Cricket」這個類別為例，調整建構子之後會變成下面這個樣子：

```
class Cricket extends Sports {

    int num_umpires;
    int num_substitutes;

    public Cricket(){
        super();
    }
}
```

由於 Cricket 的父類別「Sports」，並沒有「Sports()」這個建構子，因此原本的程式會編譯失敗。所以這題目要求替 Cricket 類別加上正確的建構子，使得程式能夠成功編譯。

選項 A 在新的建構子內的第一行加上「super(11, "Cricket", "Condidtion OK");」，呼叫原本就存在於 Sports 類別的「Sports(int np, String sname, String sground)」建構子，因此加上這段程式可以使程式編譯成功。

選項 B 的新建構子並沒有在第一行使用「super」或是「this」敘述，編譯器會自動加上「super();」，但是 Sports 類別並沒有「Sports()」這個建構子，因此一樣會編譯失敗。

選項 C 的新建構子內雖然有加入「super(11, "Cricket", "Condidtion OK");」，但它放在第二行，因此會編譯錯誤。要用「super」呼叫父類別的建構子，只能在建構子的第一行使用。

選項 D 編譯錯誤的理由和選項 C 一樣。

(218)

```
class X {

    static int i;
    int j;

    public static void main(String[] args) {
        X x1 = new X();
        X x2 = new X();
        x1.i = 3;
        x1.j = 4;
        x2.i = 5;
        x2.j = 6;
        System.out.println(
                x1.i + " "
                + x1.j + " "
                + x2.i + " "
                + x2.j);
    }
}
```

What is the result?

A
3 4 5 6

B
3 4 3 6

C(ans)
5 4 5 6

D
3 6 4 6

題解

第 7~8 行，宣告並實體化出了 x1 和 x2 這兩個 X 物件。X 物件的變數成員有物件變數 j，和宣告時用了 static 修飾的靜態(類別)變數 i。兩種變數的差別在於，類別變數只會有一個存在，其記憶體空間並不會因為類別被實體化而多產生出來，也就是說，x1 和 x2 的 i 變數是同一個變數，實際指到的記憶體空間是一樣的。

第 9~12 行，將不同的值指派給 x1 和 x2 物件的變數 i 和變數 j。由於變數 i 是類別變數，因此「x1.i」和「x2.i」都可以改寫成「X.i」。

所以到最後，X.i 為 5，而 x1.j 為 4、x2.j 為 6，輸出結果為「5 4 5 6」。

(219)

```
public class Test {
    public static void main(String[] args) {
        /* insert code here */
        array[0] = 10;
        array[1] = 20;
        System.out.print(array[0] + ":" + array[1]);
    }
}
```

Which code fragment, when inserted at line 3, enables the code to print 10:20?

A(ans)
int[] array = new int[2];

B
```
int[] array;
array = int[2];
```

C
```
int array = new int[2];
```

D
```
int array [2] ;
```

題解

題目有使用到「array」這個變數，從第 4 行到第 6 行可以看出「array」是一個陣列。原先的程式中「array」並沒有先被宣告出來，也沒有實體化任何的陣列物件給「array」變數。

選項 A 是正確的。宣告「array」變數，並實體化出一個長度為 2 的整數陣列物件。

選項 B 是把選項 A 拆成兩行，但少了 new 運算子來實體化出整數陣列物件，編譯錯誤。

選項 C 將「array」宣告成整數基本型態，又實體化出一個長度為 2 的整數陣列物件指派給「array」整數變數儲存，型態不對造成編譯錯誤。

選項 D 比較像是 C 語言宣告陣列於堆疊(stack)空間的方式，在 Java 語言中是錯誤的用法，會造成編譯錯誤。

(220)

```
public class TestTry {

    public static void main(String[] args) {
        StringBuilder message = new StringBuilder("hello
java!");
        int pos = 0;
        try {
            for (pos = 0; pos < 12; pos++) {
                switch (message.charAt(pos)) {
                    case 'a':
                    case 'e':
                    case 'o':
                        String uc =
Character.toString(message.charAt(pos)).toUpperCase();
                        message.replace(pos, pos + 1, uc);
                }
            }
        } catch (Exception e) {
            System.out.println("Out of limits");
        }
        System.out.println(message);
    }
}
```

What is the result?

A
hEllOjAvA!

B
Hello java!

C(ans)
Out of limits
hEllOjAvA!

D
Out of limits

題解
第 4 行的字串為「hello java!」，共有 11 個字元，可知字串長度是 11。
第 7 行開始的 for 迴圈，pos 變數的值為 0,1,2,3,...,11，迴圈會執行 12 次。
第 8 行開始的 switch，會在每次迴圈執行的時候會把字串中對應位置的字元「a」、「e」、「o」轉成大寫。但是當 pos 變數為 11 的時候，第 8 行會因為要取得的字元索引位置超過字串本身的長度而拋出例外，例外會在第 16 行開始被接住，而先輸出「Out of limits」。
最後，第 19 行才會輸出將「hello java!」中所有的「a」、「e」、「o」字元轉成大寫後的結果。