(1)

```
Map<Integer, String> books = new TreeMap<>();
books.put(1007, "A");
books.put(1002, "C");
books.put(1001, "B");
books.put(1003, "B");
System.out.println(books);
```

What is the result?

A
{1007 = A, 1002 = C, 1001 = B, 1003 = B}

B(ans)
{1001 = B, 1002 = C, 1003 = B, 1007 = A}

C
{1002 = C, 1003 = B, 1007 = A}

D
{1007 = A, 1001 = B, 1003 = B, 1002 = C}

題解
TreeMap 繼承 SortedMap，會自動排序元素的 key。所以這題答案是選項 B。

(2)

```
class CheckClass {

	public static int checkValue(String s1, String s2) {
		return s1.length() - s2.length();
	}
}
```

and the code fragment:

```
String[] strArray = new String[]{"Tiger", "Rat", "Cat", "Lion"};
//line n1
for (String s : strArray) {
	System.out.print(s + " ");
}
```

Which code fragment should be inserted at line n1 to enable the code to print Rat Cat Lion Tiger ?
A(ans)
Arrays.sort(strArray, CheckClass::checkValue);

B
Arrays.sort(strArray, (CheckClass::new)::checkValue);

C
Arrays.sort(strArray, (CheckClass::new).checkValue);

D
Arrays.sort(strArray, CheckClass::new::checkValue);

題解
Arrays 類別的 sort 方法可以排序傳入的陣列，也可以再傳入 Comparator 來自行決定陣列的排序方式。CheckClass 類別的 checkValue 方法是類別方法，不需要實體化即可使用，因此答案是選項 A。

(3)
```
Path source = Paths.get("/data/december/log.txt");
Path destination = Paths.get("/data");
Files.copy(source, destination);
```

and assuming that the file /data/december/log.txt is accessible and contains:

10-Dec-2014 – Executed successfully

What is the result?

A
A file with the name log.txt is created in the /data directory and the content of the /data/december/log.txt file is copied to it.

B
The program executes successfully and does NOT change the file system.

C
A FileNotFoundException is thrown at run time.

D(ans)
A FileAlreadyExistsException is thrown at run time.

題解
若/data 已存在，程式會拋出 FileAlreadyExistsException。若/data 不存在，會在「/」目錄下產生「data」檔案，內容為「10-Dec-2014 – Executed successfully」。
所以這題選擇選項 D 的話會比較合適。
若要避免 FileAlreadyExistsException，可以在 Files 類別的 copy 方法加入 CopyOption 至第三個參數中，如下：
```
Files.copy(source, destination, StandardCopyOption.REPLACE_EXISTING);
```

(4)
```
class FuelNotAvailException extends Exception {
}

class Vehicle {

    void ride() throws FuelNotAvailException { //line n1
        System.out.println("Happy Journey!");
    }
}

class SolarVehicle extends Vehicle {

    public void ride() throws Exception { //line n2
        super.ride();
    }
}
```

and the code fragment:

```
public static void main(String[] args) throws FuelNotAvailException, Exception {
      Vehicle v = new SolarVehicle();
      v.ride();
}
```

Which modification enables the code fragment to print Happy Journey!?

A
Replace line n1 with public void ride() throws FuelNotAvailException {

B(ans)
Replace line n1 with protected void ride() throws Exception {

C
Replace line n2 with void ride() throws Exception {

D
Replace line n2 with private void ride() throws FuelNotAvailException {

題解
FuelNotAvailException 繼承 Exception，屬於 checked exception。
題目原先的程式會在 line n2 編譯失敗，因為 SolarVehicle 的 ride 方法覆寫了 Vehicle 的 ride 方法，Vehicle 的 ride 方法會拋出 FuelNotAvailException，所以 SolarVehicle 的 ride 方法只能不拋出例外、拋出 unhecked exception 或是拋出 FuelNotAvailException 或其子例外。
選項 A，對編譯問題沒有幫助。
選項 B，正確。
選項 C，對編譯問題沒有幫助。
選項 D，可見度需要大於等於 default，使用 private 會編譯錯誤。

(5)

```
Path path1 = Paths.get("/app/./sys/");
Path res1 = path1.resolve("log");
Path path2 = Paths.get("/server/exe/");
Path res2 = path2.resolve("/readme/");
System.out.println(res1);
System.out.println(res2);
```

What is the result?

A
/app/sys/log
/readme/server/exe

B
/app/log/sys
/server/exe/readme

C(ans)
/app/./sys/log
/readme

D
/app/./sys/log
/server/exe/readme

題解
Path 物件的 resolve 方法類似 Paths 類別的 get 方法，可以產生新的 Path 物件。但與 Paths 類別的 get 方法不同的地方在於，Path 物件的 resolve 方法會與目前的 Path 物件有關，resolve 方法可傳入相對路徑或是絕對路徑。
程式第 13 行，為相對路徑的使用方式，路徑會在「/app/./sys」之後加上一層「log」，變成「/app/./sys/log」
程式第 15 行，為絕對路徑的使用方式，路徑為「/readme」。
所以答案是選項 C。

(6)
```
public class Book implements Comparator<Book> {

	String name;
	double price;

	public Book() {
	}

	public Book(String name, double price) {
		this.name = name;
		this.price = price;
	}

	public int compare(Book b1, Book b2) {
		return b1.name.compareTo(b2.name);
	}

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

And

```
List<Book> books = Arrays.asList(new Book("Beginning with Java", 2), new Book("A Guide to Java Tour", 3));
Collections.sort(books, new Book());
System.out.print(books);
```

What is the result?

A(ans)
[A Guide to Java Tour:3.0, Beginning with Java:2.0]

B
[Beginning with Java:2.0, A Guide to Java Tour:3.0]

C

A compilation error occurs because the Book class does not override the abstract method compareTo().

D
An Exception is thrown at run time.

題解
Book 類別實作了 Comparator 介面，並使用 Book 物件的 name 欄位的字串辭典順序來作為排序的依據。
程式第 36 行，使用了 Collections 類別的 sort 方法並傳入 Book 這個 Comparator 物件來排序集合物件。
由於 A 的辭典順序在 B 的前面，因此排序之後會輸出：
[A Guide to Java Tour:3.0, Beginning with Java:2.0]

(7)
```
public class Customer {

	private String fName;
	private String lName;
	private static int count;

	public Customer(String first, String last) {
		fName = first;
		lName = last;
		++count;
	}

	static {
		count = 0;
	}

	public static int getCount() {
		return count;
	}
}
```

App.java
```
public class App {

	public static void main(String[] args) {
		Customer c1 = new Customer("Larry", "Smith");
		Customer c2 = new Customer("Pedro", "Gonzales");
		Customer c3 = new Customer("Penny", "Jones");
		Customer c4 = new Customer("Lars", "Svenson");
		c4 = null;
		c3 = c2;
		System.out.println(Customer.getCount());
	}
}
```

What is the result?

A
0

B
2

C
3

D(ans)
4

E
5

題解
先看到 Customer 類別的 getCount 類別方法，是回傳 count 變數的值。count 變數在建構子被呼叫的時候會加 1，在 App 類別的 main 方法中，一共實體化了 4 個 Customer 物件，所以最後會輸出「4」。

(8)
```
public interface Moveable<Integer> {

	public default void walk(Integer distance) {
		System.out.println("Walking");
	}

	public void run(Integer distance);
}
```

Which statement is true?

A(ans)
Moveable can be used as below:
```
Moveable<Integer> animal = n -> System.out.println("Running" + n);
animal.run(100);
animal.walk(20);
```

B
Moveable can be used as below:
```
Moveable<Integer> animal = n -> n + 10;
animal.run(100);
animal.walk(20);
```

C
Moveable can be used as below:
```
Moveable animal = (Integer n) -> System.out.println(n);
animal.run(100);
Moveable.walk(20);
```

D
Movable cannot be used in a lambda expression.

題解
Moveable 介面需要實作的方法為僅接受傳入一個整數參數，且沒有回傳值的的 run 方法。walk 方法是 Moveable 介面的預設方法，只能在實體化出 Moveable 介面的物件後才能使用，無法直接使用 Moveable 介面來呼叫。

選項 A，正確。
選項 B，run 方法不能有回傳值。
選項 C，walk 方法不能用這樣的方式呼叫。
選項 D，Moveable 介面可以使用 Lambda 表示式來實作。

(9)

```
public class Counter {

	public static void main(String[] args) {
		int a = 10;
		int b = -1;
		assert (b >= 1) : "Invalid Denominator";
		int c = a / b;
		System.out.println(c);
	}
}
```

What is the result of running the code with the -ea option?

A
-10

B
0

C(ans)
An AssertionError is thrown.

D
A compilation error occurs.

題解
Java 的 assertion 敘述通常用於 private 方法內，確保傳入方法的參數範圍。當 assertion 敘述的判斷式為 false 的時候，就會拋出 AssertionError。在執行 Java 程式的時候，若要啟用 assertion 敘述的功能，必須要在「java」指令後加上「-ea」或是「-enableassertions」參數來啟用 assertion。
在這題的程式中，assertion 敘述的判斷式為 false，所以會拋出 AssertionError。

(10)

```
class Vehicle {

	String name;

	void setName(String name) {
		this.name = name;
	}

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

Which action encapsulates the Vehicle class?

A
Make the Vehicle class public.

B
Make the name variable public.

C
Make the setName method public.

D(ans)
Make the name variable private.

E
Make the setName method private.

F
Make the getName method private.

題解
封裝的概念在於保護程式碼，以及物件的欄位資料的正確性。
選項 A，與封裝的概念無關。
選項 B，與封裝的概念背道而馳。應將 name 欄位設成 private 可見度後，使用 getter 和 setter 來存取 name 欄位的資料。
選項 C，setName 方法是不是 public 都與封裝概念沒有太大的關聯。
選項 D，正確。
選項 E，setName 方法是不是 private 都與封裝概念沒有太大的關聯。
選項 F，getName 方法是不是 private 都與封裝概念沒有太大的關聯。

(11)
```
class RateOfInterest {

	public static void main(String[] args) {
		int rateOfInterest = 0;
		String accountType = "LOAN";
		switch (accountType) {
			case "RD":
				rateOfInterest = 5;
				break;
			case "FD":
				rateOfInterest = 10;
				break;
			default:
				assert false : "No interest for this account"; //line n1
		}
		System.out.println("Rate of interest:" + rateOfInterest);
	}
}
```

and the command:
```
java -ea RateOfInterest
```

What is the result?

A

Rate of interest: 0

B(ans)
An AssertionError is thrown.

C
No interest for this account

D
A compilation error occurs at line n1.

題解
Java 的 assertion 敘述通常用於 private 方法內，確保傳入方法的參數範圍。當 assertion 敘述的判斷式為 false 的時候，就會拋出 AssertionError。在執行 Java 程式的時候，若要啟用 assertion 敘述的功能，必須要在「java」指令後加上「-ea」或是「-enableassertions」參數來啟用 assertion。
在這題的程式中，accountType 所參考到的字串均不符合 swicth 結構中的 case 項目，因此會執行第 22 行 default 的程式。所以第 23 行的 assertion 敘述執行的時候，就會拋出 AssertionError。

(12)
```
List<String> empDetails = Arrays.asList("100, Robin, HR",
            "200, Mary, AdminServices",
            "101, Peter, HR");
empDetails.stream()
            .filter(s -> s.contains("1"))
            .sorted()
            .forEach(System.out::println); //line n1
```

What is the result?

A(ans)
100, Robin, HR
101, Peter, HR

B
A compilation error occurs at line n1

C
100, Robin, HR
101, Peter, HR
200, Mary, AdminServices

D
100, Robin, HR
200, Mary, AdminServices
101, Peter, HR

題解
串流物件的 filter 方法可以保留符合條件的元素，因此在程式第 24 行之後，串流物件中的元素剩下「100, Robin, HR」和「101, Peter, HR」。sorted 方法會排序串流中的元素，若不指定 Comparator 來排序字串元素，會使用辭典順序來排序，第 25 行排序之後的順序依然是「100, Robin, HR」在「101, Peter, HR」之前。

程式第 26 行使用串流的 forEach 方法將每個元素輸出至標準輸出串流中，所以答案是選項 A。

(13)

```
Path p1 = Paths.get("/Pics/MyPic.jpeg");
System.out.println(p1.getNameCount()
            + ":" + p1.getName(1)
            + ":" + p1.getFileName());
```

Assume that the Pics directory does NOT exist.
What is the result?

A
An exception is thrown at run time.

B(ans)
2:MyPic.jpeg:MyPic.jpeg

C
1:Pics:/Pics/MyPic.jpeg

D
2:Pics:MyPic.jpeg

題解
Path 物件的 getNameCount 方法會計算路徑結構中檔案或是目錄名稱的數量，在這個題目中的路徑為「/Pics/MyPic.jpeg」，因此有「Pics」和「MyPic.jpeg」兩個名稱。
Path 物件的 getName 方法會取得路徑結構中的檔案或是目錄名稱，索引值從 0 開始。
Path 物件的 getFileName 方法會取得最後一個名稱。
因此這題不論檔案或是目錄存不存在，都會輸出：
2:MyPic.jpeg:MyPic.jpeg

(14)

```
List<String> nL = Arrays.asList("Jim", "John", "Jeff");
Function<String, String> funVal = s -> "Hello : ".concat(s);
nL.stream()
            .map(funVal)
            .peek(System.out::print);
```

What is the result?

A
Hello : JimHello : JohnHello : Jeff

B
JimJohnJeff

C(ans)
The program prints nothing.

D
A compilation error occurs.

題解
這個題目提供的程式碼並不會輸出任何東西，若將第 22 行開始的程式修改如下：

```
List<String> nL = Arrays.asList("Jim", "John", "Jeff");
Function<String, String> funVal = s -> "Hello : ".concat(s);
nL.stream()
            .map(funVal)
            .forEach(System.out::print);
```
或：
```
nL.stream()
            .map(funVal)
            .peek(System.out::print)
            .count();
```
才會輸出：
Hello : JimHello : JohnHello : Jeff

(15)
```
Connection conn = DriverManager.getConnection(dbURL, userName, passWord);
String query = "SELECT id FROM Employee";
try (Statement stmt = conn.createStatement()) {
      ResultSet rs = stmt.executeQuery(query);
      stmt.executeQuery("SELECT id FROM Customer");
      while (rs.next()) {
            //process the results
            System.out.println("Employee ID: " + rs.getInt("id"));
      }
} catch (Exception e) {
      System.out.println("Error");
}
```
Assume that:
The required database driver is configured in the classpath.
The appropriate database is accessible with the dbURL, userName, and passWord exists.
The Employee and Customer tables are available and each table has id column with a few records and the SQL queries are valid.

What is the result of compiling and executing this code fragment?

A
The program prints employee IDs.

B
The program prints customer IDs.

C(ans)
The program prints Error.

D
compilation fails on line 13.

題解
程式第 13 行，又用了一次 Statement 物件的 executeQuery 方法，會使得先前的 ResultSet 自動被關閉，因此程式第 14 行會拋出 SQLException

(16)
```
Stream<Path> files = Files.walk(Paths.get(System.getProperty("user.home")));
files.forEach(fName -> { //line n1
      try {
            Path aPath = fName.toAbsolutePath(); //line n2
```

```
            System.out.println(fName + ":"
                        +                                   Files.readAttributes(aPath,
BasicFileAttributes.class).creationTime());
      } catch (IOException ex) {
            ex.printStackTrace();
      }
});
```

What is the result?

A(ans)
All files and directories under the home directory are listed along with their attributes.

B
A compilation error occurs at line n1.

C
The files in the home directory are listed along with their attributes.

D
A compilation error occurs at line n2.

題解
「System.getProperty("user.home")」會取得使用者的家目錄路徑。Paths 類別的 get 方法可以傳入路徑字串物件或是 URI 物件來產生 Path 物件。Files 類別的 walk 方法可以產生串流物件，以深度優先走訪傳入的 Path 物件所代表的檔案目錄。
所以選項 A 是正確答案。

(17)
```
Stream<List<String>> iStr = Stream.of(
            Arrays.asList("1", "John"),
            Arrays.asList("2", null));
IntStream nInSt = iStr.flatMapToInt((x) -> x.stream());
nInSt.forEach(System.out::print);
```

What is the result?

A
1John2null

B
12

C
A NullPointerException is thrown at run time.

D(ans)
A compilation error occurs.

題解
iStr 所參考到的串流物件中的元素為字串型態的 List 物件，無法直接轉換成整數型態的 List 物件。若修改成以下程式，可以成功編譯：
IntStream nInSt = iStr.flatMapToInt((x) -> x.stream().mapToInt(Integer::valueOf));

以上程式雖然可以成功編譯，但執行時會拋出 NumberFormatException，因為「John」字串不能轉成整數。
如果要能成功執行，不應使用 flatMapToInt 方法，應改寫程式如下：
IntStream nInSt = iStr.mapToInt((x) -> Integer.parseInt(x.get(0)));
改寫之後程式會輸出：
12

(18)

```
public class Emp {

	private String eName;
	private Integer eAge;

	Emp(String eN, Integer eA) {
		this.eName = eN;
		this.eAge = eA;
	}

	public Integer getEAge() {
		return eAge;
	}

	public String getEName() {
		return eName;
	}
}
```

and code fragment:

```
List<Emp> li = Arrays.asList(new Emp("Sam", 20), new Emp("John", 60), new Emp("Jim", 51));
Predicate<Emp> agVal = s -> s.getEAge() > 50; //line n1
li = li.stream().filter(agVal).collect(Collectors.toList());
Stream<String> names = li.stream().map(Emp::getEName); //line n2
names.forEach(n -> System.out.print(n + " "));
```

What is the result?

A
Sam John Jim

B(ans)
John Jim

C
A compilation error occurs at line n1.

D
A compilation error occurs at line n2.

題解
程式第 32 行，串流物件的 filter 方法可以保留符合條件的元素，collect 方法可以將串流物件再轉成別的 Collection 物件，配合 Collectors 類別的 toList 方法可以將串流物件中的元素清單化，轉成 List 物件。至於元素在新的 List 物件中的順序就是原先

集合物件的走訪順序，所以「Emp("John", 60)」在「Emp("Jim", 51)」之前。所以轉出來的 List 物件元素為，「Emp("John", 60), Emp("Jim", 51)」。
程式第 33 行，使用串流物件的 map 方法取代原本的元素。
程式第 34 行，輸出「John Jim 」。

(19)

```
class Sum extends RecursiveAction { //line n1

	static final int THRESHOLD_SIZE = 3;
	int stIndex, lstIndex;
	int[] data;

	public Sum(int[] data, int start, int end) {
		this.data = data;
		this.stIndex = start;
		this.lstIndex = end;
	}

	protected void compute() {
		int sum = 0;
		if (lstIndex - stIndex <= THRESHOLD_SIZE) {
			for (int i = stIndex; i < lstIndex; i++) {
				sum += data[i];
			}
			System.out.println(sum);
		} else {
			new Sum(data, stIndex + THRESHOLD_SIZE, lstIndex).fork();
			new Sum(data, stIndex,
					Math.min(lstIndex, stIndex + THRESHOLD_SIZE)
			).compute();
		}
	}
}
```

and the code fragment:

```
ForkJoinPool fjPool = new ForkJoinPool();
int data[] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
fjPool.invoke(new Sum(data, 0, data.length));
```

and given that the sum of all integers from 1 to 10 is 55.

Which statement is true?

A(ans)
The program prints several values that total 55.

B
The program prints 55.

C
A compilation error occurs at line n1.

D
The program prints several values whose sum exceeds 55.

題解

ForkJoinPool 可以將一個大問題拆成很多小問題來進行運算。而 RecursiveAction 是一種 ForkJoinTask，沒有回傳值。
在這個題目中，要利用 ForkJoinPool 和 RecursiveAction 來進行整數陣列的元素加總計算。
呼叫 ForkJoinPool 物件的 invoke 方法後，會執行 RecursiveAction 的 compute 方法，此時 stIndex=0、lstIndex=10。程式第 15 行的判斷式不成立，會再 fork 出新的 Sum 物件，用新的執行緒來計算「stIndex=3、lstIndex=10」的部份，而目前的執行緒則重新計算「stIndex=0、lstIndex=3」的部份。
「stIndex=0、lstIndex=3」，會計算 sum=1+2+3=6，輸出「6」。
「stIndex=3、lstIndex=10」，會再 fork 出「stIndex=6、lstIndex=10」，計算「stIndex=3、lstIndex=6」的 sum 為 4+5+6=15，輸出「15」。
「stIndex=6、lstIndex=10」，會再 fork 出「stIndex=9、lstIndex=10」，計算「stIndex=6、lstIndex=9」的 sum 為 7+8+9=24，輸出「24」。
「stIndex=9、lstIndex=10」，會計算 sum=10，輸出「10」。
以上輸出由於是在不同執行緒處理的關係，所以順序不是固定的，但這些輸出的值加總後，即為 1 加到 10 的總和「55」。所以答案是選項 A。

(20)

USCurrency.java

```
public enum USCurrency {
	PENNY(1),
	NICKLE(5),
	DIME(10),
	QUARTER(25);
	private int value;

	public USCurrency(int value) {
		this.value = value;
	}

	public int getValue() {
		return value;
	}
}
```

Coin.java

```
public class Coin {

	public static void main(String[] args) {
		USCurrency usCoin = new USCurrency.DIME;
		System.out.println(usCoin.getValue());
	}
}
```

Which two modifications enable the given code to compile?

A
Nest the USCurrency enumeration declaration within the Coin class.

B(ans)
Make the USCurrency enumeration constructor private.

C(ans)

Remove the new keyword from the instantion of usCoin.

D
Make the getter method of value as a static method.

E
Add the final keyword in the declaration of value.

題解
enum 建構子的可見度只能使用 default 或是 private。使用 enum 預先列舉好的常數不需要使用 new 運算子。因此這題答案是選項 B 和選項 C

(21)
Path file = Paths.get("courses.txt");
// line n1
Assume the courses.txt is accessible.
Which code fragment can be inserted at line n1 to enable the code to print the content of the courses.txt file?

A
List<String> fc = Files.list(file);
fc.stream().forEach (s -> System.out.println(s));

B
Stream<String> fc = Files.readAllLines(file);
fc.forEach (s -> System.out.println(s));

C
List<String> fc = readAllLines(file);
fc.stream().forEach (s -> System.out.println(s));

D(ans)
Stream<String> fc = Files.lines(file);
fc.forEach (s -> System.out.println(s));

題解
選項 A，Files 類別的 list 方法會回傳 Stream 物件，所以會編譯錯誤。
選項 B，Files 類別的 readAllLines 方法會回傳 List 物件，所以會編譯錯誤。
選項 C，readAllLines 方法前應該指定 Files 類別。
選項 D，正確答案。

(22)
String str = "Java is a programming language";
ToIntFunction<String> indexVal = str::indexOf; //line n1
int x = indexVal.applyAsInt("Java"); //line n2
System.out.println(x);

What is the result?

A(ans)
0

B
1

C
A compilation error occurs at line n1.

D
A compilation error occurs at line n2.

題解
ToIntFunction 的 JDK 原始碼如下：
```
@FunctionalInterface
public interface ToIntFunction<T> {

	int applyAsInt(T value);
}
```
所以 line n2，其實會等於：
```
int x = str.indexOf("Java");
```
會在 str 所參考到的字串中，索引 0 的位置找到「Java」子字串。

(23)
Which statement is true about java.util.stream.Stream?

A
A stream cannot be consumed more than once.

B(ans)
The execution mode of streams can be changed during processing.

C
Streams are intended to modify the source data.

D
A parallel stream is always faster than an equivalent sequential stream.

題解
選項 A，串流是可以被消耗很多次的，例如使用 peak 方法。
選項 B，這個選項應該是指串流的序列處理模式和平行化處理模式。
選項 C，串流並不會修改到原始的資料，而是產生新的資料出來。
選項 D，平行化的串流並不總是比序列的串流快，因為多執行緒會需要消耗額外的資源。

(24)
```
List<String> listVal = Arrays.asList("Joe", "Paul", "Alice", "Tom");
System.out.println(
	// line n1
);
```

Which code fragment, when inserted at line n1, enables the code to print the count of string elements whose length is greater than three?

A(ans)
listVal.stream().filter(x -> x.length() > 3).count()

B
listVal.stream().map(x -> x.length() > 3).count()

C
listVal.stream().peek(x -> x.length() > 3).count().get()

D
listVal.stream().filter(x -> x.length() > 3).mapToInt(x -> x).count()

題解
Stream 物件的 filter 方法可以保留符合條件的元素；map 和 mapToInt 方法可以用來替換元素；peek 方法可以用來走訪所有元素。題目要求找出字串長度大於三的元素數量。
選項 A，用 filter 方法只保留字串長度大於三的元素，所以剩下來的元素數量為 2。是正確答案。
選項 B，用 map 方法將所有元素轉為 boolean，數量依然為 4。
選項 C，peak 方法的用法不正確。
選項 D，mapToInt 方法的用法不正確。

(25)
```
public void recDelete(String dirName) throws IOException {
	File[] listOfFiles = new File(dirName).listFiles();
	if (listOfFiles != null && listOfFiles.length > 0) {
		for (File aFile : listOfFiles) {
			if (aFile.isDirectory()) {
				recDelete(aFile.getAbsolutePath());
			} else if (aFile.getName().endsWith(".class")) {
				aFile.delete();
			}
		}
	}
}
```

Assume that Projects contains subdirectories that contain .class files and is passed as an argument to the recDelete() method when it is invoked.
What is the result?

A(ans)
The method deletes all the .class files in the Projects directory and its subdirectories.

B
The method deletes the .class files of the Projects directory only.

C
The method executes and does not make any changes to the Projects directory.

D
The method throws an IOException.

題解
程式第 16 行，會找出 dirName 這個目錄下的所有檔案(不包括子目錄中的檔案)。程式第 18 行的 for 迴圈，會走訪所有檔案。若檔案為目錄，則再用這個目錄遞迴呼叫 recDelete 方法；若檔案不是目錄，且檔名是「.class」結尾的話，則刪除之。
所以答案是選項 A。

(26)

```
class Employee {

	Optional<Address> address;

	Employee(Optional<Address> address) {
		this.address = address;
	}

	public Optional<Address> getAddress() {
		return address;
	}
}

class Address {

	String city = "New York";

	public String getCity() {
		return city;
	}

	public String toString() {
		return city;
	}
}
```

And

```
Address address = null;
Optional<Address> addrs1 = Optional.ofNullable(address);
Employee e1 = new Employee(addrs1);
String eAddress = (addrs1.isPresent()) ? addrs1.get().getCity() : "City Not available";
System.out.println(eAddress);
```

What is the result?

A
New York

B(ans)
City Not available

C
Null

D
A NoSuchElementException is thrown at run time.

題解

Optional 類別的 ofNullable 方法允許傳入參考為 null 的物件參數，來產生 Optional 物件。由於第 41 行的 address 變數為 null，因此會產生出空的 Optional 物件。
程式第 42 行，利用空的 Optional 物件來實體化新的 Employee 物件，這個沒有問題。
程式第 43 行，先判斷 Optional 物件的內容是否存在，由於這裡使用的 Optional 物件是空的，因此會將「City Not available」字串參考指派給 eAddress 變數儲存。

最後輸出「City Not available」。
若在程式有執行到「addrs1.get()」，將會因為 addrs1 所參考到的 Optional 物件的內容是空的，而拋出 NoSuchElementException 例外。但在這個題目中，使用了三元條件運算子來控制程式的流程，所以不會執行到「addrs1.get()」。

(27)

```
public class Foo {

	public static void main(String[] args) {
		Map<Integer, String> unsortMap = new HashMap<>();
		unsortMap.put(10, "z");
		unsortMap.put(5, "b");
		unsortMap.put(1, "d");
		unsortMap.put(7, "e");
		unsortMap.put(50, "j");
		Map<Integer, String> treeMap = new TreeMap<>(new Comparator<Integer>()
{
			@Override
			public int compare(Integer o1, Integer o2) {
				return o2.compareTo(o1);
			}
		});
		treeMap.putAll(unsortMap);
		for (Map.Entry<Integer, String> entry : treeMap.entrySet()) {
			System.out.print(entry.getValue() + " ");
		}
	}
}
```

What is the result?

A
A compilation error occurs.

B
d b e z j

C(ans)
j z e b d

D
z b d e j

題解
TreeMap 實作了 SortedMap 介面，所以存在 TreeMap 的項目元素是經過排序的。在實體化 TreeMap 物件的時候可以傳入自訂的 Comparator 物件來決定項目元素間比較大小的方式，但排序的依據是項目的 key，而不是 value。在這個題目中型態為 Integer 的 key，比較大小的方式為反序排列，也就是說，數值愈大的會愈在前面。例如：1 會在 0 之前，2 會在 1 之前。
所以這題答案為選項 C。

(28)

```
class Worker extends Thread {
```

```
	CyclicBarrier cb;

	public Worker(CyclicBarrier cb) {
		this.cb = cb;
	}

	public void run() {
		try {
			cb.await();
			System.out.println("Worker...");
		} catch (Exception ex) {
		}
	}
}

class Master implements Runnable { //line n1

	public void run() {
		System.out.println("Master...");
	}
}
```

and the code fragment:

```
Master master = new Master();
//line n2
Worker worker = new Worker(cb);
worker.start();
```

You have been asked to ensure that the run methods of both the Worker and Master classes are executed.

Which modification meets the requirement?

A
At line n2, insert CyclicBarrier cb = new CyclicBarrier(2, master);

B
Replace line n1 with class Master extends Thread {

C(ans)
At line n2, insert CyclicBarrier cb = new CyclicBarrier(1, master);

D
At line n2, insert CyclicBarrier cb = new CyclicBarrier(master);

題解
程式第 42 行，在 Worker 的建構子用到了 cb 變數作為引數傳入，所以 line n2 一定要定義 cb 變數是什麼東西。
CyclicBarrier 是 Java 內建的類別，位於 java.util.concurrent 套件下，用來讓某個執行緒等待至指定數量的執行緒呼叫 CyclicBarrier 的 await 方法後才繼續執行。
選項 A，此種 CyclicBarrier 物件實體化方式需要呼叫兩次 CyclicBarrier 的 await 方法後，await 方法之後的程式才會被執行，但是在題目給的程式中，await 只有在第 11 行被呼叫過，因此永遠執行不到第 12 行，而且也無法執行 master 這個 Runnable 物件。
選項 B，應該要修改 line n2 才是正確的。

選項 C，此種 CyclicBarrier 物件實體化方式只需要呼叫一次 CyclicBarrier 的 await 方法後，傳入 CyclicBarrier 建構子的 master 這個 Runnable 物件和 await 方法之後的程式都會被執行，所以這是正確答案。
選項 D，CyclicBarrier 類別沒有這種建構子。

(29)

```
class Student {

	String course, name, city;

	public Student(String name, String course, String city) {
		this.course = course;
		this.name = name;
		this.city = city;
	}

	public String toString() {
		return course + ":" + name + ":" + city;
	}

	public String getCourse() {
		return course;
	}
}
```

and the code fragment:

```
List<Student> stds = Arrays.asList(
		new Student("Jessy", "Java ME", "Chicago"),
		new Student("Helen", "Java EE", "Houston"),
		new Student("Mark", "Java ME", "Chicago"));
stds.stream()
		.collect(Collectors.groupingBy(Student::getCourse))
		.forEach((src, res) -> System.out.println(src));
```

What is the result?
A
[Java EE: Helen:Houston]
[Java ME: Jessy:Chicago, Java ME: Mark:Chicago]

B(ans)
Java EE
Java ME

C
[Java ME: Jessy:Chicago, Java ME: Mark:Chicago]
[Java EE: Helen:Houston]

D
A compilation error occurs.

題解
程式第 45 行，串流物件的 collect 方法可以將串流物件再轉成別的 Collection 物件，配合 Collectors 類別的 groupingBy 方法可以將串流物件中的元素群組化，轉成 Map 物件。群組化的依據為串流物件的 Country 物件元素之呼叫 getCourse 方法後的回傳內容，這個會作為 Map 的 Key 值。至於 Map 的 Key 值所對應的元素，即為原先的物件。

因此最後產生的 Map 集合物件，「Java ME」這個課程有「Java ME:Jessy:Chicago」和「Java ME:Mark:Chicago」，而「Java EE」這個課程下只有「Java EE:Helen:Houston」。至於元素在 Map 物件中的順序，愈先建立群組的項目會排在愈後面，所以「Java ME」會排在「Java EE」之後。元素在 List 物件中的順序就是原先集合物件的走訪順序，所以「Java ME:Jessy:Chicago」在「Java ME:Mark:Chicago」之前。
程式第 46 行，只會輸出 Map 集合物件的 key 值，因此答案是選項 B。

(30)

```
class Bird {

	public void fly() {
		System.out.print("Can fly");
	}
}

class Penguin extends Bird {

	public void fly() {
		System.out.print("Cannot fly");
	}
}
```

and the code fragment:

```
class Birdie {

	public static void main(String[] args) {
		fly(() -> new Bird());
		fly(Penguin::new);
	}
	/* line n1 */
}
```

Which code fragment, when inserted at line n1, enables the Birdie class to compile?

A

```
static void fly (Consumer<Bird> bird) {
	bird::fly();
}
```

B

```
static void fly (Consumer<? extends Bird> bird) {
	bird.accept().fly();
}
```

C(ans)

```
static void fly (Supplier<Bird> bird) {
	bird.get().fly();
}
```

D

```
static void fly (Supplier<? extends Bird> bird) {
	bird::fly();
}
```

題解

以下分別是 Consumer 和 Supplier 介面於 JDK 的原始碼片段：

```
@FunctionalInterface
public interface Consumer<T> {
```
Consumer.java
```
	void accept(T t);
}
```
Supplier.java
```
@FunctionalInterface
public interface Supplier<T> {

	T get();
}
```
從原始碼可以很清楚知道：Consumer 介面的 accept 方法只能從參數接受物件，不能回傳物件；而 Supplier 介面的 get 方法只能回傳物件，不能接受參數。所以選項 C 是正確答案。

(31)
```
abstract class Shape {
	Shape() { System.out.println("Shape"); }
	protected void area() { System.out.println("Shape"); }
}

class Square extends Shape {
	int side;
	Square(int side) {
		/* insert code here */
		this.side = side;
	}
	public void area() { System.out.println("Square"); }
}
class Rectangle extends Square {
	int len, br;
	Rectangle(int x, int y) {
		/* insert code here */
		len = x; br = y;
	}
	void area() { System.out.println("Rectangle"); }
}
```
Which two modifications enable the code to compile?

A
At line 1, remove abstract

B
At line 9, insert super();

C
At line 12, remove public

D(ans)
At line 17, insert super(x);

E
At line 17, insert super(); super.side = x;

F(ans)

At line 20, use public void area () {

題解
原先題目的程式會在第 16 行和第 20 行編譯錯誤，原因在於 Rectangle 類別的建構子沒有指定要先呼叫哪個建構子，因此會自動加上「super();」去呼叫 Square 類別沒有參數的建構子，但這個建構子並不存在。而 Rectangle 類別中的 area 方法，覆寫了 Square 類別中的 area 方法，但是可見度卻從 public 降到 default，這是不被允許的。
所以說，選項 D 和選項 F 是正確的。
選項 C 不行的原因在於，Shape 類別的 area 方法之可見度為 protected，所以 Square 類別的 area 方法之可見度必定也不能低於 protected。

(32)
```
public static void main(String[] args) throws IOException {
      BufferedReader brCopy = null;
      try (BufferedReader br = new BufferedReader(new FileReader("employee.txt")))
{ // line n1
            br.lines().forEach(c -> System.out.println(c));
            brCopy = br;//line n2
      }
      brCopy.ready(); //line n3;
}
```
Assume that the ready method of the BufferedReader, when called on a closed BufferedReader, throws an exception, and employee.txt is accessible and contains valid text.
What is the result?

A
A compilation error occurs at line n3.

B
A compilation error occurs at line n1.

C
A compilation error occurs at line n2.

D(ans)
The code prints the content of the employee.txt file and throws an exception at line n3.

題解
br 變數所參考到的 BufferedReader 物件為 try-with-resources 結構的資源，因此在離開 try-with-resources 結構的 try 區塊後，資源就會被釋放(BufferedReader 物件會 close)。line n3 的部份在 BufferedReader 物件被 close 之後呼叫 ready 方法，所以會拋出 IOException。

(33)
Which statement is true about the single abstract method of the java.util.function.Function interface?

A
It accepts one argument and returns void.

B
It accepts one argument and returns boolean.

C
It accepts one argument and always produces a result of the same type as the argument.

D(ans)
It accepts an argument and produces a result of any data type.

題解
Function 的 JDK 原始碼片段如下：
```
public interface Function<T, R> {
	R apply(T t);
}
```
選項 A，要回傳 R 型態的物件才對。
選項 B，要回傳 R 型態的物件才對。
選項 C，傳入的參數型態不一定要跟回傳型態一樣。
選項 D，正確。

(34)
```
void doStuff() throws ArithmeticException, NumberFormatException, Exception {
   if (Math.random() > -1) throw new Exception ("Try again");
}
```
And
```
try {
	doStuff();
} catch (ArithmeticException | NumberFormatException | Exception e) {
	System.out.println(e.getMessage());
} catch (Exception e) {
	System.out.println(e.getMessage());
}
```
Which modification enables the code to print Try again?

A
Comment the lines 28, 29 and 30.

B
Replace line 26 with:
} catch (Exception | ArithmeticException | NumberFormatException e) {

C(ans)
Replace line 26 with:
} catch (ArithmeticException | NumberFormatException e) {

D
Replace line 27 with:
throw e;

題解
題目原先提供的程式會在第 26 行和第 28 行編譯錯誤。
第 26 行錯誤的原因在於，ArithmeticException 和 NumberFormatException 均為 Exception 的子類別，重複包含了。
第 28 行錯誤的原因在於，第 26 行已經有 catch 到 Exception 了，所以永遠執行不到第 26 行。
要修正這個問題，可以把第 26 行的 Exception 拿掉，因此答案是選項 C。

(35)

```
interface Doable {

	public void doSomething(String s);
}
```

Which two class definitions compile?

A(ans)

```
public abstract class Task implements Doable {

	public void doSomethingElse(String s) {
	}
}
```

B

```
public abstract class Work implements Doable {

	public abstract void doSomething(String s) {
	}

	public void doYourThing(Boolean b) {
	}
}
```

C

```
public class Job implements Doable {

	public void doSomething(Integer i) {
	}
}
```

D

```
public class Action implements Doable {

	public void doSomething(Integer i) {
	}

	public String doThis(Integer j) {
	}
}
```

E(ans)

```
public class Do implements Doable {

	public void doSomething(Integer i) {
	}

	public void doSomething(String s) {
	}

	public void doThat(String s) {
	}
}
```

題解

選項 A，雖然 Task 類別沒有實作出 Doable 介面的 doSomething 方法，但是 Task 類別為抽象類別，因此允許未實作的方法。
選項 B，抽象方法不能有程式實作區塊。
選項 C，Job 類別的 doSomething 抽象方法傳入的參數和 Doable 介面的 doSomething 方法不同，將不被認為是實作 Doable 介面的 doSomething 方法，所以 Job 類別會因沒有實作 Doable 介面的 doSomething 方法而無法編譯。
選項 D，錯誤理由同選項 C。
選項 E，有成功實作出 Doable 介面的 doSomething 方法。

(36)

```
ZonedDateTime depart = ZonedDateTime.of(2015, 1, 15, 3, 0, 0, 0, ZoneId.of("UTC-7"));
ZonedDateTime arrive = ZonedDateTime.of(2015, 1, 15, 9, 0, 0, 0, ZoneId.of("UTC-5"));
long hrs = ChronoUnit.HOURS.between(depart, arrive); //line n1
System.out.println("Travel time is " + hrs + " hours");
```

What is the result?

A(ans)
Travel time is 4 hours

B
Travel time is 6 hours

C
Travel time is 8 hours

D
An exception is thrown at line n1.

題解
這題是在考 Java 8 加入的日期與時間(Date-Time)API
ChronoUnit 類別可以利用 Duration 類別計算兩日期或是時間的間隔。
(2015/1/15 9 + 5) - (2015/1/15 3 + 7) = 4 hours
所以答案是選項 A。

(37)

```
class MyThread implements Runnable {

	private static AtomicInteger count = new AtomicInteger(0);

	public void run() {
		int x = count.incrementAndGet();
		System.out.print(x + " ");
	}
}
```

and

```
Thread thread1 = new Thread(new MyThread());
Thread thread2 = new Thread(new MyThread());
Thread thread3 = new Thread(new MyThread());
Thread[] ta = {thread1, thread2, thread3};
for (int x = 0; x < 3; x++) {
	ta[x].start();
}
```

Which statement is true?

A(ans)
The program prints 1 2 3 and the order is unpredictable.

B
The program prints 1 2 3.

C
The program prints 1 1 1.

D
A compilation error occurs.

題解
AtomicInteger 的 incrementAndGet 方法，可以使 AtomicInteger 所表示的整數值加一後，同時回傳整數結果，在操作 AtomicInteger 物件的數值時，同一時間永遠只有一個執行緒可以存取，所以可以達成 thread-safe。
程式第 8 行的 x 區域變數，在每個執行緒中的數值都不一樣，排序後為 1、2、3。但是執行緒執行第 9 行的順序不同，因此輸出結果會是 1、2、3 的任意排列組合。

(38)
```
public class Book {

	private String read(String bname) {
		return "Read" + bname;
	}
}

public class EBook extends Book {

	public String read(String url) {
		return "View" + url;
	}
}

public class Test {

	public static void main(String[] args) {
		Book b1 = new Book();
		b1.read("Java Programing");
		Book b2 = new EBook();
		b2.read("http://ebook.com/ebook");
	}
}
```
What is the result?

A
Read Java Programming
View http://ebook.com/ebook

B
Read Java Programming
Read http://ebook.com/ebook

C

The EBook.java file fails to compile.

D(ans)
The Test.java file fails to compile.

題解
Test 類別會編譯失敗，因為 Book 物件的 read 方法是 private 可見度，第 5 行和第 7 行都會編譯失敗。

(39)
```
public class Product {

	int id;
	int price;

	public Product(int id, int price) {
		this.id = id;
		this.price = price;
	}

	public String toString() {
		return id + " : " + price;
	}
}
```

and the code fragment:
```
List<Product> products = new ArrayList(Arrays.asList(new Product(1, 10),
		new Product(2, 30),
		new Product(2, 30)));
Product p = products.stream().reduce(new Product(4, 0), (p1, p2) -> {
	p1.price += p2.price;
	return new Product(p1.id, p1.price);
});
products.add(p);
products.stream().parallel()
		.reduce((p1, p2) -> p1.price > p2.price ? p1 : p2)
		.ifPresent(System.out::println);
```
What is the result?

A
2 : 30

B
4 : 0

C(ans)
4 : 70

D
4 : 70
2 : 30
3 : 20
1 : 10

E

The program prints nothing.

題解
程式第 33 行，reduce 方法會將串流中的所有 Product 物件元素的 price 欄位進行加總，10+30+30=70，最後得到的 Product 物件為「4：70」。
程式第 39 行，reduce 方法會保留最後 price 欄位數值最大的 Product 物件，因此會把「4：70」留下。
程式第 40 行，輸出「4：70」。

(40)
```
List<String> colors = Arrays.asList("red", "green", "yellow");
Predicate<String> test = n -> {
      System.out.println("Searching...");
      return n.contains("red");
};
colors.stream()
            .filter(c -> c.length() > 3)
```

What is the result?

A(ans)
Searching...

B
Searching...
Searching...

C
Searching...
Searching...
Searching...

D
A compilation error occurs.

題解
程式第 12 行的 filter 方法只會保留包含字串長度大於 3 的字串元素。只有「green」和「yellow」符合條件。
程式第 13 行利用 allMatch 方法來判斷串流中的字串元素是否都包含「red」子字串，若找到一個不符合條件的字串元素，就會立即回傳 false。因此在這個題目中，只會輸出一次「Searching...」。

(41)
Given the content of /resourses/Message.properties:
`welcome1="Good day!"`
and given the code fragment:

```
Properties prop = new Properties();
FileInputStream fis = new FileInputStream("/resources/Message.properties");
prop.load(fis);
System.out.println(prop.getProperty("welcome1"));
System.out.println(prop.getProperty("welcome2", "Test"));//line n1
System.out.println(prop.getProperty("welcome3"));
```

What is the result?

A
Good day!
Test
followed by an Exception stack trace

B
Good day!
followed by an Exception stack trace

C(ans)
Good day!
Test
null

D
A compilation error occurs at line n1.

題解
程式第 13 行，會輸出「/resourses/Message.properties」中「welcome1」鍵值所對應的值，即為「Good day!」。
程式第 14 行，由於「/resourses/Message.properties」中不存在「welcome2」鍵值，但有設定預設值「Test」，所以會輸出預設值「Test」。
程式第 15 行，由於「/resourses/Message.properties」中不存在「welcome3」鍵值，所以輸出「null」。

(42)
```
LocalDate valentinesDay = LocalDate.of(2015, Month.FEBRUARY, 14);
LocalDate nextYear = valentinesDay.plusYears(1);
nextYear.plusDays(15); //line n1
System.out.println(nextYear);
```

What is the result?

A(ans)
2016-02-14

B
A DateTimeException is thrown.

C
2016-02-29

D
A compilation error occurs at line n1.

題解
這題是在考 Java 8 加入的日期與時間(Date-Time)API
程式第 6 行會建立出一個「2015/02/14」的 LocalDate 物件。
程式第 7 行會建立出一個「2016/02/14」的 LocalDate 物件。
程式第 8 行會建立出一個「2016/02/29」的 LocalDate 物件，但這個物件參考不會被任何變數儲存。

程式第 9 行會輸出第 7 行 nextYear 變數所參考到的 LocalDate 物件，也就是「2016/02/14」，所以會輸出「2016-02-14」。

(43)
Which statement is true about the DriverManager class?

A(ans)
It returns an instance of Connection.

B
It executes SQL statements against the database.

C
It only queries metadata of the database.

D
It is written by different vendors for their specific database.

題解
選項 A，使用 DriverManager 類別的 getConnection，可以連接資料庫並取得 Connection 物件實體。
選項 B，Statement 物件才可以執行 SQL 敘述。
選項 C，DriverManager 是用來載入 JDBC 的 Driver 和連接資料庫。
選項 D，這個敘述描述的應是 Driver 類別。

(44)
Which two reasons should you use interfaces instead of abstract classes?

A
You expect that classes that implement your interfaces have many common methods or fields, or require access modifiers other than public.

B(ans)
You expect that unrelated classes would implement your interfaces.

C
You want to share code among several closely related classes.

D
You want to declare non-static on non-final fields.

E(ans)
You want to take advantage of multiple inheritance of type.

題解
選項 A，介面定義的成員一定是 public 可見度，所以這不是正確答案。
選項 B，如果說 A 繼承 B 是「A 是 B(A is B)」的關係，那麼 A 實作 B 可以說是「A 能做到 B 能做的事(A is capable of doing what B can do)」。也就是說，介面比較像是提供物件功能方面的描述，例如有個介面叫作「Walkable」(能走)，定義了「walk」(走路)的方法，然後有個「Human」(人)這個類別，實作了「Walkable」介面。「Human」和「Walkable」並沒有什麼直接的關係，任何類別都可以實作這個介面來表示類別能做到走路這個動作。因此這個選項的敘述是正確的。
選項 C，介面除了特殊的預設方法和靜態方法用法之外，是沒辦法共享程式實作的。

選項 D，介面的欄位一定會使用 public static final 來進行修飾。
選項 E，一個類別可以實作多個介面。

(45)
Which statement is true about java.time.Duration?

A
It tracks time zones.

B
It preserves daylight saving time.

C(ans)
It defines time-based values.

D
It defines date-based values.

題解
選項 A，Duration 和時區沒有關係。
選項 B，Duration 也跟日光節約時間沒有關係。
選項 C，正確，Duration 是以時間為基礎。
選項 D，以日期為基礎的應該為 Period。

(46)
Which action can be used to load a database driver by using JDBC3.0?

A
Add the driver class to the META-INF/services folder of the JAR file

B
Include the JDBC driver class in a jdbc.properties file.

C(ans)
Use the java.lang.Class.forName method to load the driver class.

D
Use the DriverManager.getDriver method to load the driver class.

題解
JDBC4.0 之前，載入 Driver 的方式就是使用 Class 類別的 forName 方法，在參數指定 Driver 的名稱(類別路徑)，程式會自動呼叫 DriverManager 類別的 registerDriver 方法來載入(註冊)Driver。
選項 A，這是 JDBC4.0 的機制，要將 Driver 的名稱寫進「META-INF/services/java.sql.Driver」檔案內，即可自動加載。
選項 B，沒有這樣的用法。
選項 C，正確載入 Driver 的方式。
選項 D，DriverManager 類別的 getDriver 方法只是用來取得已經載入的 Driver 物件。

(47)
Given the records from the Employee table:
eid ename

111 Tom
112 Jerry
113 Donald

and given the code fragment:

```
try {
	Connection conn = DriverManager.getConnection(URL, userName, passWord);
	Statement st = conn.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,
ResultSet.CONCUR_UPDATABLE);
	st.execute("SELECT * FROM Employee");
	ResultSet rs = st.getResultSet();
	while (rs.next()) {
		if (rs.getInt(1) == 112) {
			rs.updateString(2, "Jack");
		}
	}
	rs.absolute(2);
	System.out.println(rs.getInt(1) + " " + rs.getString(2));
} catch (SQLException ex) {
	System.out.println("Exception is raised");
}
```

Assume that:

The required database driver is configured in the classpath.
The appropriate database accessible with the URL, userName, and passWordexists.

What is the result?

A(ans)
The Employee table is updated with the row:
112 Jack
and the program prints:
112 Jerry

B
The Employee table is updated with the row:
112 Jack
and the program prints:

C
The Employee table is not updated and the program prints:
112 Jerry

D
The program prints Exception is raised.

題解

第 22 行建立 Statement 物件時所傳入的 resultSetType 為 TYPE_SCROLL_INSENSITIVE，因此除了可以用 next 來移動欄位指標之外，也可以使用 afterLast、previous、absolute 和 relative 等方法。
TYPE_SCROLL_INSENSITIVE 會保留原始從資料庫取得的資料，
TYPE_SCROLL_SENSITIVE 才會使用後來更動的資料。
while 迴圈會執行三次，走訪完所有的結果列，此時 eid 為 112 的列的第二個欄位，原先的「Jerry」字串已被修改為「Jack」字串。但由於 resultSetType 為 TYPE_SCROLL_INSENSITIVE，會保留一開始的結果，所以最後輸出「112 Jerry」。

(48)

```
class Vehicle {

	int vno;
	String name;

	public Vehicle(int vno, String name) {
		this.vno = vno;
		this.name = name;
	}

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

and this code fragment:

```
Set<Vehicle> vehicles = new TreeSet<>();
vehicles.add(new Vehicle(10123, "Ford"));
vehicles.add(new Vehicle(10124, "BMW"));
System.out.println(vehicles);
```

What is the result?

A
10123:Ford
10124:BMW

B
10124:BMW
10123:Ford

C
A compilation error occurs.

D(ans)
A ClassCastException is thrown at run time.

題解
題目提供的程式會在第 21 行拋出 ClassCastException，因為 TreeSet 的元素必須要實作 Comparable 介面，才可以進行自動排序。

(49)

```
Student (id INTEGER, name VARCHAR)

public class Test {

	static Connection newConnection = null;

	public static Connection getDBConnection() throws SQLException {
		try (Connection con = DriverManager.getConnection(URL, username,
password)) {
			newConnection = con;
		}
		return newConnection;
```

```
    }

    public static void main(String[] args) throws SQLException {
        getDBConnection();
        Statement st = newConnection.createStatement();
        st.executeUpdate("INSERT INTO student VALUES (102, 'Kelvin')");
    }
}
```

Assume that:

The required database driver is configured in the classpath.
The appropriate database is accessible with the URL, userName, and passWord exists.
The SQL query is valid.

What is the result?

A
The program executes successfully and the STUDENT table is updated with one record.

B
The program executes successfully and the STUDENT table is NOT updated with any record.

C(ans)
A SQLException is thrown as runtime.

D
A NullPointerException is thrown as runtime.

題解

Connection 物件是 try-with-resources 結構的資源，因此在離開 try-with-resources 結構的 try 區塊時會自動呼叫其 close 方法。由於 Connection 已經 close，所以在第 14 行，嘗試使用 Connection 物件來建立 Statement 物件時，會拋出 SQLException。

(50)

```
class CallerThread implements Callable<String> {

    String str;

    public CallerThread(String s) {
        this.str = s;
    }

    public String call() throws Exception {
        return str.concat(" Call");
    }
}
```

And

```
public static void main(String[] args) throws InterruptedException, ExecutionException
{
    ExecutorService es = Executors.newFixedThreadPool(4); //line n1
    Future f1 = es.submit(new CallerThread("Call"));
    String str = f1.get().toString();
    System.out.println(str);
}
```

Which statement is true?

A
The program prints Call Call and terminates.

B(ans)
The program prints Call Call and does not terminate.

C
A compilation error occurs at line n1.

D
An ExecutionException is thrown at run time.

題解
程式第 32 行，實體化出 CallerThread 的物件並透過 ExecutorService 來執行。
程式第 33 行，會等待 CallerThread 物件的 call 方法執行完畢後，將結果回傳，儲存至 str 變數中。回傳的物件為「Call Call」字串。
程式第 34 行，輸出「Call Call」，
由於沒有呼叫 ExecutorService 物件的 shutdown 相關方法，因此程式不會停止執行。

(51)
```
List<Integer> list1 = Arrays.asList(10, 20);
List<Integer> list2 = Arrays.asList(15, 30);
//line n1
```

Which code fragment, when inserted at line n1, prints 10 20 15 30?

A(ans)
```
Stream.of(list1, list2)
.flatMap(list -> list.stream())
.forEach(s -> System.out.print(s + " "));
```

B
```
Stream.of(list1, list2)
.flatMap(list -> list.intStream())
.forEach(s -> System.out.print(s + " "));
```

C
```
list1.stream()
.flatMap(list2.stream().flatMap(e1 -> e1.stream())
.forEach(s -> System.out.println(s + " "));
```

D
```
Stream.of(list1, list2)
.flatMapToInt(list -> list.stream())
.forEach(s -> System.out.print(s + " "));
```

題解
Stream 物件的 flatMap 方法可以將串流物件中的元素展開成新的 Stream 物件。
選項 A，正確答案。
選項 B，Stream 物件並沒有 intStream 方法。
選項 C，e1 的型態是整數，所以沒有 stream 方法。

選項 D，Stream 物件的 flatMapToInt 方法，可以將串流物件中的元素展開成新的 IntStream 物件。所以傳入的 mapper 參數的 Lambda 語法，應該要能回傳 IntStream 物件才行。如果改成如以下程式，則選項 D 正確：

```
Stream.of(list1, list2)
.flatMapToInt(list -> list.stream().mapToInt(e -> e))
.forEach(s -> System.out.print(s + " "));
```

(52)

Canvas.java

```
public class Canvas implements Drawable {

	public void draw() {
	}
}
```

Board.java

```
public abstract class Board extends Canvas {
}
```

Paper.java

```
public class Paper extends Canvas {

	protected void draw(int color) {
	}
}
```

Frame.java

```
public class Frame extends Canvas implements Drawable {

	public void resize() {
	}
}
```

Drawable.java

```
public interface Drawable {

	public abstract void draw();
}
```

Which statement is true?

A
Board does not compile.

B
Paper does not compile.

C
Frame does not compile.

D
Drawable does not compile.

E(ans)
All classes compile successfully.

(53)

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

```
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        System.out.print("Enter GDP: ");
        //line 1
}
```

Which code fragment, when inserted at line 1, enables the code to read the GDP from the user?

A(ans)

```
int GDP = Integer.parseInt(br.readline());
```

B

```
int GDP = br.read();
```

C

```
int GDP = br.nextInt();
```

D

```
int GDP = Integer.parseInt(br.next());
```

題解

選項 A，從標準輸入串流讀取一行字串之後，轉成整數儲存。

選項 B，BufferedReader 物件的 read 方法會讀取一個字元，接著回傳字元的值，不是題目想要的結果。

選項 C，BufferedReader 物件沒有 nextInt 方法，會編譯錯誤。Scanner 物件才有。

選項 D，BufferedReader 物件沒有 next 方法，會編譯錯誤。Scanner 物件才有。

(54)

```
List<String> str = Arrays.asList("my", "pen", "is", "your", "pen");
Predicate<String> test = s -> {
        int i = 0;
        boolean result = s.contains("pen");
        System.out.print(i++ + " : ");
        return result;
};
str.stream()
            .filter(test)
            .findFirst()
            .ifPresent(System.out::print);
```

What is the result?

A(ans)

0 : 0 : pen

B

0 : 1 : pen

C

0 : 0 : 0 : 0 : 0 : pen

D

0 : 1 : 2 : 3 : 4 :

E

A compilation error occurs.

題解
程式第 18 行的 filter 方法只會保留包含「pen」子字串的字串元素。第 14 行的「i++」可以直接看成 0，因為 i 是區域變數，所以每次取值都會是 0。程式第 19 行使用了 findFirst 方法，會使得第 18 行的 filter 方法找到第一個符合「保留包含『pen』子字串的字串元素」後，就直接回傳結果。因此程式會找到索引 1 的「pen」，然後將「pen」字串物件包成 Optional 後回傳，此時程式輸出「0 : 0 :」。第 19 行的 ifPresent 方法，因為 Optional 物件中有「pen」字串的存在，因此會執行標準輸出串流物件的 print 方法，輸出「pen」。

(55)
```
UnaryOperator<Integer> uo1 = s -> s * 2; //line n1
List<Double> loanValues = Arrays.asList(1000.0, 2000.0);
loanValues.stream()
            .filter(lv -> lv >= 1500)
            .map(lv -> uo1.apply(lv)) //line n2
            .forEach(s -> System.out.print(s + " "));
```
What is the result?

A
4000.0

B
4000

C
A compilation error occurs at line n1.

D(ans)
A compilation error occurs at line n2.

題解
程式第 10 行會編譯錯誤，原因在於 List 的泛型型態為 Double，而 UnaryOperator 卻是使用 Integer，兩者不同所以編譯錯誤。

(56)
You have been asked to create a ResourceBundle which uses a properties file to localize an application.

Which code example specifies valid keys of menu1 and menu2 with values of File Menu and View Menu?

A
File Menu
View Menu

B
menu1File Menu
menu2View Menu

C
menu1, File Menu, menu2, View Menu

D(ans)

menu1 = File Menu
menu2 = View Menu

題解
ResourceBundle 是使用「key=value」的結構來存放資料。

(57)

You want to create a singleton class by using the Singleton design pattern.

Which two statements enforce the singleton nature of the design?

A
Make the class static.

B(ans)
Make the constructor private.

C
Override equals() and hashCode() methods of the java.lang.Object class.

D(ams)
Use a static reference to point to the single instance.

E
Implement the Serializable interface.
E. Implement the Serializable interface.

題解
Singleton design pattern 能保證一個類別只能有一個物件實體。
選項 A，靜態類別和它能產生多少個實體並沒有關聯。
選項 B，用 private 來修飾建構子，可以有效地阻決外部類別任意實體化出物件實體。
選項 C，equals 和 hashCode 方法和物件實體化沒有什麼關聯。
選項 D，用靜態的變數來儲存第一次實體化出來的物件參考，接下來都返回這個物件參考即可。
選項 E，Serializable 和實體化沒有什麼關聯。

(57)
IceCream.java
```
public final class IceCream {

	public void prepare() {
	}
}
```
Cake.java
```
public class Cake {

	public final void bake(int min, int temp) {
	}

	public void mix() {
	}
}
```

Shop.java

```
public class Shop {

	private Cake c = new Cake();
	private final double discount = 0.25;

	public void makeReady() {
		c.bake(10, 120);
	}
}
```

Bread.java

```
public class Bread extends Cake {

	public void bake(int minutes, int temperature) {
	}

	public void addToppings() {
	}
}
```

Which statement is true?

A
A compilation error occurs in IceCream.

B
A compilation error occurs in Cake.

C
A compilation error occurs in Shop.

E(ans)
A compilation error occurs in Bread

E
All classes compile successfully.

題解
Bread.java 第三行的 bake 方法會編譯錯誤，因為 Bread 類別繼承的 Cake 類別，其 bake 方法已經使用了 final 來修飾，無法被覆寫。

(58)

```
interface Rideable {

	Car getCar(String name);
}

class Car {

	private String name;

	public Car(String name) {
		this.name = name;
	}
}
```

Which code fragment creates an instance of Car?

A
```
Car auto = Car("MyCar")::new;
```

B
```
Car auto = Car::new;
Car vehicle = auto::getCar("MyCar");
```

C(ans)
```
Rideable rider = Car::new;
Car vehicle = rider.getCar("MyCar");
```

D
```
Car vehicle = Rideable::new::getCar("MyCar");
```

題解
選項 A，錯誤的用法。
選項 B，錯誤的用法。Car 並不是 Function Interface，無法直接指派 Lambda 語法。
選項 C，正確的用法。用 Lambda 語法來實作出 Rideable 介面的 getCar 方法，在這裡使用 Lambda 精簡語法的方式省略了以下匿名類別程式：
```
Rideable rider = new Rideable() {
	@Override
	public Car getCar(String name) {
		return new Car(name);
	}
};
```
或 Lambda 語法：
```
Rideable rider = name -> new Car(name);
```
如此一來，Rideable 物件的 getCar(String)方法就可以當作是 new Car(String)來使用。
選項 D，錯誤的用法。

(59)
```
public class Emp {

	String fName;
	String lName;

	public Emp(String fn, String ln) {
		fName = fn;
		lName = ln;
	}

	public String getfName() {
		return fName;
	}

	public String getlName() {
		return lName;
	}
}
and the code fragment:
List<Emp> emp = Arrays.asList(
		new Emp("John", "Smith"),
		new Emp("Peter", "Sam"),
```

```
            new Emp("Thomas", "Wale"));
emp.stream()
            //line n1
            .collect(Collectors.toList());
```

Which code fragment, when inserted at line n1, sorts the employees list in descending order of fName and then ascending order of lName?

A(ans)
.sorted(Comparator.comparing(Emp::getfName).reversed().thenComparing(Emp::getlName))

B
.sorted(Comparator.comparing(Emp::getfName).thenComparing(Emp::getlName))

C
.map(Emp::getfName).sorted(Comparator.reverseOrder())

D
.map(Emp::getfName).sorted(Comparator.reserveOrder().map(Emp::getlName).reserved

題解
選項 A，正確答案。
選項 B，fName 和 lName 都是遞增排序，不合題目要求。
選項 C，用 map 方法會替換掉原本的元素，所以會只剩下遞減排序的 fName。
選項 D，編譯錯誤，串流在經過第一次 map 方法之後，其元素就是字串物件而不是 Emp 物件了

(60)

```
List<Integer> codes = Arrays.asList(10, 20);
UnaryOperator<Double> uo = s -> s + 10.0;
codes.replaceAll(uo);
codes.forEach(c -> System.out.println(c));
```

What is the result?

A
20.0
30.0

B
10
20

C(ans)
A compilation error occurs.

D
A NumberFormatException is thrown at run time.

題解
程式第 12 行會編譯錯誤，原因在於 List 的泛型型態為 Integer，而 UnaryOperator 卻是使用 Double，兩者不同所以編譯錯誤。

(61)

```
final class Folder { //line n1

	//line n2

	public void open() {
		System.out.print("Open");
	}
}

public class Test {

	public static void main(String[] args) throws Exception {
		try (Folder f = new Folder()) {
			f.open();
		}
	}
}
```

Which two modifications enable the code to print Open Close?

A(ans)
Replace line n1 with:
class Folder implements AutoCloseable {

B
Replace line n1 with:
class Folder extends Closeable {

C
Replace line n1 with:
class Folder extends Exception {

D
At line n2, insert:

```
final void close() {
	System.out.print("Close");
}
```

E(ans)
At line n2, insert:

```
public void close() throws IOException {
	System.out.print("Close");
}
```

題解
要把物件當作 try-with-resources 的資源，該物件類別必須要實作 AutoCloseable 介面的 close 方法。
AutoCloseable 介面的程式碼如下：

```
public interface AutoCloseable {
	void close() throws Exception;
}
```

AutoCloseable 介面中定義的 close 方法有拋出 Exception，所以實作 close 方法時也允許拋出 Exception。雖然這裡 close 方法並沒有使用 public 修飾字來修飾，但因為 AutoCloseable 是介面，成員都會自動加上 public 修飾字，因此實作 close 方法時也要需要 public 修飾字來修飾。

所以答案是選項 A 和選項 E。

(62)
For which three objects must a vendor provide implementations in its JDBC driver?

A
Time

B
Date

C(ans)
Statement

D(ans)
ResultSet

E(ans)
Connection

F
SQLException

G
DriverManager

題解
必須實作 Connection、Statement、PreparedStatement、CallableStatement、ResultSet 和 Driver。

(63)

```
class ImageScanner implements AutoCloseable {

	public void close() throws Exception {
		System.out.print("Scanner closed. ");
	}

	public void scanImage() throws Exception {
		System.out.print("Scan.");
		throw new Exception("Unable to scan. ");
	}
}

class ImagePrinter implements AutoCloseable {

	public void close() throws Exception {
		System.out.print("Printer closed. ");
	}

	public void printImage() {
		System.out.print("Print.");
	}
}
```

and this code fragment:

```
try (ImageScanner ir = new ImageScanner();
            ImagePrinter iw = new ImagePrinter()) {
      ir.scanImage();
      iw.printImage();
} catch (Exception e) {
      System.out.print(e.getMessage());
}
```

What is the result?

A(ans)

Scan.Printer closed. Scanner closed. Unable to scan.

B

Scan.Scanner closed. Unable to scan.

C

Scan. Unable to scan.

D

Scan. Unable to scan. Printer closed.

題解

程式第 32 行，呼叫 ImageScanner 物件的 scanImage 方法，會先輸出「Scan.」然後再拋出訊息為「Unable to scan. 」的 Exception，這個 Exception 在 try-with-resources 會被接住，在離開 try 的區塊進入 catch 區塊前，會先自動呼叫 ImagePrinter 物件的 call 方法，輸出「Printer closed. 」，再自動呼叫 ImageScanner 物件的 call 方法，輸出「Scanner closed. 」。因為 ImageScanner 物件比 ImagePrinter 物件還要早宣告與實體化出來，基於 try-with-resources 結構中的資源為先進後出的特性，ImagePrinter 物件會比 ImageScanner 物件還要早被關閉。

在 catch 區塊中，會輸出「Unable to scan. 」。

(64)

```
class UserException extends Exception {
}

class AgeOutOfLimitException extends UserException {
}
```

and the code fragment:

```
class App {

      public void doRegister(String name, int age) throws UserException,
AgeOutOfLimitException {
            if (name.length() < 6) {
                  throw new UserException();
            } else if (age >= 60) {
                  throw new AgeOutOfLimitException();
            } else {
                  System.out.println("User is registered.");
            }
      }

      public static void main(String[] args) throws UserException {
            App t = new App();
            t.doRegister("Mathew", 60);
      }
```

}
What is the result?

A
User is registered.

B(ans)
An AgeOutOfLimitException is thrown.

C
A UserException is thrown.

D
A compilation error occurs in the main method

題解
由於第 23 行呼叫 App 物件的 doRegister 方法所傳入的 age 參數為 60，因此第 14 行的 if 條件式會成立，所以在執行第 15 行時拋出 AgeOutOfLimitException。

(65)
```
IntStream stream = IntStream.of(1, 2, 3);
IntFunction<Integer> inFu = x -> y -> x * y; // line n1
IntStream newStream = stream.map(inFu.apply(10)); // line n2
newStream.forEach(System.out::print);
```

A
Replace line n1 with:
```
IntFunction<UnaryOperator> inFu = x -> y -> x * y;
```

B(ans)
Replace line n1 with:
```
IntFunction<IntUnaryOperator> inFu = x -> y -> x * y;
```

C
Replace line n1 with:
```
BiFunction<IntUnaryOperator> inFu = x -> y -> x * y;
```

D
Replace line n2 with:
```
BiFunction<IntUnaryOperator> inFu = x -> y -> x * y;
```

題解
這題原先的程式在 line n1 和 line n2 都會編譯錯誤，原因在於 inFu 變數的泛型型態不是 IntUnaryOperator，line n1 的 Lambda 語法回傳形態有誤，line n2 的 apply 方法之傳入參數型態有誤。所以答案是選項 B。

(66)
```
List<String> codes = Arrays.asList("DOC", "MPEG", "JPEG");
codes.forEach(c -> System.out.print(c + " "));
String fmt = codes.stream()
            .filter(s -> s.contains("PEG"))
            .reduce((s, t) -> s + t).get();
System.out.println("\n" + fmt);
```

What is the result?

A(ans)
DOC MPEG JPEG
MPEGJPEG

B
DOC MPEG MPEGJPEG
MPEGMPEGJPEG

C
MPEGJPEG
MPEGJPEG

D
The order of the output is unpredictable.

題解
程式第 6 行，用 Collection 物件的 forEach 方法來走訪所有集合的元素，輸出「DOC MPEG JPEG 」。
程式第 8 行，使用串流物件的 filter 方法，只保留串流物件中包含「PEG」子字串的字串元素，所以只剩下「MPEG」和「JPEG」。
程式第 9 行，使用串流物件的 reduce 方法，可以將串列的所有元素縮減成一個元素，在這裡縮減的方式為直接串接字串，會串接成「MPEGJPEG」。最後使用 get 方法來取得字串物件。

(67)
```
public class Test<T> {

	private T t;

	public T get() {
		return t;
	}

	public void set(T t) {
		this.t = t;
	}

	public static void main(String args[]) {
		Test<String> type = new Test<>();
		Test type1 = new Test(); //line n1
		type.set("Java");
		type1.set(100); //line n2
		System.out.print(type.get() + " " + type1.get());
	}
}
```
What is the result?

A(ans)
Java 100

B
java.lang.string@java.lang.Integer@

C
A compilation error occurs. To rectify it, replace line n1 with:
`Test<Integer> type1 = new Test<>();`

D
A compilation error occurs. To rectify it, replace line n2 with:
`Test<Integer> type1 = new Test<>();`

題解
type 變數有使用到泛型，指定的型態為 String，所以 T 為 String。type1 變數則沒有指定泛型，所以 T 可當作是 Object。
type1 物件的 get 方法，可以傳入型態為 Object 的參數，因此所有型態的資料都可以傳入，當然也包括整數物件(Integer)了。
「+」運算子若其中一個的運算元為字串(物件)時，作為字串連接的功用。物件會使用其「toString」方法來取得字串。所以這題會輸出「Java 100」。

(68)
```
public class FileThread implements Runnable {

	String fName;

	public FileThread(String fName) {
		this.fName = fName;
	}

	public void run() {
		System.out.println(fName);
	}

	public static void main(String[] args) throws IOException, InterruptedException {
		ExecutorService executor = Executors.newCachedThreadPool();
		Stream<Path> listOfFiles = Files.walk(Paths.get("Java Projects"));
		listOfFiles.forEach(line -> {
			executor.execute(new	FileThread(line.getFileName().toString()));
//line n1
		});
		executor.shutdown();
		executor.awaitTermination(5, TimeUnit.DAYS); //line n2
	}
}
```
The Java Projects directory exists and contains a list of files.
What is the result?

A
The program throws a runtime exception at line n2.

B(ans)
The program prints files names concurrently.

C
The program prints files names sequentially.

D

A compilation error occurs at line n1.

題解
Files 類別的 walk 方法可以使用深度優先(depth-first)來走訪傳入的檔案路徑。
ExecutorService 物件的 execute 方法可以使用新的執行緒來執行傳入的 Runnable 物件。shutdown 方法可以關閉 ExecutorService 物件使其不再接受新的執行物件(Runnable 或是 Callable)，但並不會等待目前正在執行和正在排隊的工作執行完畢。awaitTermination 方法可以指定一個上限時間來等待 ExecutorService 物件將所有的工作處理完成。
所以這題答案是選項 B。

(69)
```
class TechName {

	String techName;

	TechName(String techName) {
		this.techName = techName;
	}
}
```
And
```
List<TechName> tech = Arrays.asList(
		new TechName("Java-"),
		new TechName("Oracle DB-"),
		new TechName("J2EE-")
);
Stream<TechName> stre = tech.stream();
//line n1
```

Which should be inserted at line n1 to print Java-Oracle DB-J2EE-?

A
```
stre.forEach(System.out::print);
```

B(ans)
stre.map(a -> a.techName).forEach(System.out::print);

C
```
stre.map(a -> a).forEachOrdered(System.out::print);
```

D
```
stre.forEachOrdered(System.out::print);
```

題解
選項 A，由於 TechName 類別沒有覆寫 toString 方法，因此無法直接輸出 techName 欄位的值，而是輸出最上層 Object 類別的 toString 方法所產生出來的字串。
選項 B，利用 Stream 物件的 map 方法去替換 Stream 物件的元素為 TechName 物件的 techName 欄位值，之後再使用 forEach 方法將 techName 欄位值輸出，可得到題目要的輸出結果。
選項 C，無法直接輸出 techName 欄位的值，理由同選項 A。可以改成以下程式，即可得到正確結果：
```
stre.map(a -> a.techName).forEachOrdered(System.out::print);
```
選項 D，無法直接輸出 techName 欄位的值，理由同選項 A。

(70)

```
class Caller implements Callable<String> {

	String str;

	public Caller(String s) {
		this.str = s;
	}

	public String call() throws Exception {
		return str.concat(" Caller");
	}
}

class Runner implements Runnable {

	String str;

	public Runner(String s) {
		this.str = s;
	}

	public void run() {
		System.out.println(str.concat(" Runner"));
	}
}
```

And

```
public static void main(String[] args) throws InterruptedException, ExecutionException
{
	ExecutorService es = Executors.newFixedThreadPool(2);
	Future f1 = es.submit(new Caller("Call"));
	Future f2 = es.submit(new Runner("Run"));
	String str1 = (String) f1.get();
	String str2 = (String) f2.get();//line n1
	System.out.println(str1 + " : " + str2);
}
```

What is the result?

A(ans)

The program prints:
Run Runner
Call Caller : null
And the program does not terminate.

B

The program terminates after printing:
Run Runner
Call Caller : Run

C

A compilation error occurs at line n1.

D

An Execution is thrown at run time.

題解
程式第 52 行，會建立最多可以擁有兩個執行緒的 ExecutorService。
程式第 53 行，會建立 Caller 物件，並使用新的執行緒去執行 Caller 物件的 call 方法。
程式第 54 行，會建立 Runnable 物件，並使用新的執行緒去執行 Runnable 物件的 run 方法。
程式第 55 行，會等待呼叫 Caller 物件的 call 方法的執行緒是否已經執行完畢，接著回傳 call 方法的回傳值。
程式第 56 行，會等待呼叫 Runnable 物件的 run 方法的執行緒是否已經執行完畢，接著回傳 null。
在程式第 54 行至 56 行之間，某條 ExecutorService 的執行緒會執行到 Runnable 物件的 run 方法內的第 23 行，輸出「Run Runner」。
程式第 57 行，輸出「Call Caller : null」。

(71)
Which two statements are true about localizing an application?

A(ans)
Support for new regional languages does not require recompilation of the code.

B
Textual elements (messages and GUI labels) are hard-coded in the code.

C
Language and region-specific programs are created using localized data.

D
Resource bundle files include data and currency information.

E(ans)
Language codes use lowercase letters and region codes use uppercase letters.

題解
選項 A，正確。
選項 B，文字是從外部動態讀入才對，如透過 ResourceBundle 類別。
選項 C，不正確，敘述應改為國際化(internationalized)的程式。
選項 D，ResourceBundle 的檔案是包含 Locale 的訊息才對。
選項 E，正確。例如 en-US(英文，美國)、zh-TW(中文，台灣)。

(72)
```
List<Integer> values = Arrays.asList(1, 2, 3);
values.stream()
        .map(n -> n * 2) //line n1
        .peek(System.out::print) //line n2
        .count();
```

What is the result?

A(ans)
246

B
The code produces no output.

C
A compilation error occurs at line n1.

D
A compilation error occurs at line n2.

題解
串流物件的 map 方法可以取得並取代串流元素的內容。程式第 7 行會用 Lambda 語法將所有元素的值乘上 2。
串流物件的 peek 方法可以取得所有串流元素的內容，類似 forEach 方法，但 peek 方法會回傳原本的串流物件。程式第 8 行會用 Lambda 的精簡語法呼叫標準輸出串流的 print 方法，依序輸出「246」。
程式第 9 行用了 count 方法來計算元素數量，若第 8 行使用 forEach 方法而不是 peek 方法的話，第 9 行會編譯失敗。

(73)
```
BiFunction<Integer, Double, Integer> val = (t1, t2) -> t1 + t2; // line n1
System.out.println(val.apply(10, 10.5));
```

What is the result?

A
20

B
20.5

C(ans)
A compilation error occurs at line n1.

D
A compilation error occurs at line n2.

題解
BiFunction 為可以接受兩個參數的 Functional Interface，擷取 JDK 的程式碼如下：
```
public interface BiFunction<T, U, R> {
        R apply(T t, U u);
}
```
這題的 line n1 會發生編譯錯誤，因為 BiFunction 的回傳泛型型態為 Integer，但是第二個參數 U 的泛型型態為 Double。若直接把 Integer 物件加上 Double 物件或是 int 加上 double，出來的結果都會是 double，double 無法隱含式(implicit)轉型為 Integer。
若使用顯式(explicit)轉型的話，可以解決這個問題，如下：
```
BiFunction<Integer, Double, Integer> val = (t1, t2) -> (int) (t1 + t2); // line n1
System.out.println(val.apply(10, 10.5));
```
如此一來，輸出結果就會是：
20

(74)
Given that course.txt is accessible and contains:
Course::Java

and given the code fragment:

```
public static void main(String[] args){
	int i;
	char c;
	try (FileInputStream fis = new FileInputStream("/home/magiclencourse.txt");
			InputStreamReader isr = new InputStreamReader(fis);) {
		while (isr.ready()) { //line n1
			isr.skip(2);
			i = isr.read();
			c = (char) i;
			System.out.print(c);
		}
	} catch (Exception e) {
		e.printStackTrace();
	}
}
```

What is the result?

A
ur :: va

B(ans)
ueJa

C
The program prints nothing.

D
A compilation error occurs at line n1.

題解
用「*」來表示目前檔案指標的位置。
Course::Java
*
第 11 行 whlie 迴圈條件式成立。執行第 12 行，指標往後跳兩格。
Course::Java
  *
執行第 13 行，讀取到「u」之後，指標往後跳一格。
Course::Java
   *
執行第 15 行，輸出「u」。接著回到程式第 11 行，whlie 迴圈條件式成立，會執行第 12 行，指標往後跳兩格。
Course::Java
      *
執行第 13 行，讀取到「e」之後，指標往後跳一格。
Course::Java
       *
執行第 15 行，輸出「e」。接著回到程式第 11 行，whlie 迴圈條件式成立，會執行第 12 行，指標往後跳兩格。
Course::Java
          *
執行第 13 行，讀取到「J」之後，指標往後跳一格。
Course::Java
           *

執行第 15 行，輸出「J」。接著回到程式第 11 行，whlie 迴圈條件式成立，會執行第 12 行，指標往後跳兩格。
Course::Java
*
執行第 13 行，讀取到「a」之後，指標往後跳一格。
Course::Java
*
執行第 15 行，輸出「a」。接著回到程式第 11 行，已經讀取到檔案結尾，所以 whlie 迴圈條件式不成立，跳出 while 迴圈。

(75)
```
List<Integer> nums = Arrays.asList(10, 20, 8);
System.out.println(
            //line n1
    );
```
Which code fragment must be inserted at line n1 to enable the code to print the maximum number in the nums list?

A(ans)
```
nums.stream().max(Comparator.comparing(a -> a)).get()
```

B
```
nums.stream().max(Integer::max).get()
```

C
```
nums.stream().max()
```

D
```
nums.stream().map(a -> a).max()
```

題解
串流物件的 max 方法可以取得串流元素中的最大值，需要提供 Comparator 物件作為比較數值大小的依據。
選項 A，使用 Comparator 類別的 comparing 方法來直接指定要使用元素的哪個值來當作比較大小的依據。
選項 B，Comparator 是使用 0 為基準，依靠大於 0、小於 0、等於 0 的判斷來表示兩數的大小。Integer 的 Max 方法，會比較傳入的兩個參數，回傳較大的那個，並不是我們要的方法。應該要改成使用 compare 方法才正確，如下：
```
nums.stream().max(Integer::compare).get()
```
選項 C，max 方法的用法錯誤。
選項 D，map 方法是用來快速替換元素的內容，而使用「a -> a」替換後的元素就是之前的元素。在這裡若要正確使用 max 方法，可改寫如下：
```
nums.stream().map(a -> a).max(Integer::compare).get()
```

(76)
Which two code blocks correctly initialize a Locale variable?

A
```
Locale loc1 = "UK";
```

B
```
Locale loc2 = Locale.getInstance("ru");
```

C
Locale loc3 = Locale.getLocaleFactory("RU");

D(ans)
Locale loc3 = Locale.getLocaleFactory("RU");

E(ans)
Locale loc3 = Locale.getLocaleFactory("RU");

題解
選項 A，字串「UK」不能直接自動轉型成 Locale 物件。
選項 B，Locale 類別沒有 getInstance 方法。
選項 C，Locale 類別沒有 getLocaleFactory 方法。
選項 D，正確。
選項 E，正確。

(77)
```
public class Country {

	public enum Continent {
		ASIA, EUROPE
	}
	String name;
	Continent region;

	public Country(String na, Continent reg) {
		name = na;
		region = reg;
	}

	public String getName() {
		return name;
	}

	public Continent getRegion() {
		return region;
	}
}
```
and the code fragment:
```
List<Country> couList = Arrays.asList(
		new Country("Japan", Country.Continent.ASIA),
		new Country("Italy", Country.Continent.EUROPE), new
Country("Germany", Country.Continent.EUROPE));
Map<Country.Continent, List<String>> regionNames = couList.stream().
		collect(Collectors.groupingBy(Country::getRegion,
				Collectors.mapping(Country::getName, Collectors.toList())));
System.out.println(regionNames);
```

What is the output?

A(ans)
{EUROPE = [Italy, Germany], ASIA = [Japan]}

B
{ASIA = [Japan], EUROPE = [Italy, Germany]}

C
{EUROPE = [Germany, Italy], ASIA = [Japan]}

D
{EUROPE = [Germany], EUROPE = [Italy], ASIA = [Japan]}

題解
程式第 43 行，取得 couList 清單物件的串流物件。
程式第 44 和 45 行，串流物件的 collect 方法可以將串流物件再轉成別的 Collection 物件，配合 Collectors 類別的 groupingBy 方法可以將串流物件中的元素群組化，轉成 Map 物件。群組化的依據為串流物件的 Country 物件元素之呼叫 getRegion 方法後的回傳內容，這個會作為 Map 的 Key 值。至於 Map 的 Key 值所對應的元素，在這裡使用了 Collectors 類別的 mapping 方法，將串流物件的 Country 物件元素之呼叫 getName 方法後的回傳內容轉成 List 物件。
因此最後產生的 Map 集合物件，「ASIA」這個地區下只有「Japan」，而「EUROPE」這個地區下有「Italy」和「Germany」。至於元素在 Map 物件中的順序，愈先建立群組的項目會排在愈後面，所以「ASIA」會排在「EUROPE」之後。元素在 List 物件中的順序就是原先集合物件的走訪順序，所以「Italy」在「Germany」之前。

(78)
Given:
Item table
ID, INTEGER: PK
DESCRIP, VARCHAR(100)
PRICE, REAL
QUANTITY, INTEGER
And given the code fragment:
```
try {
	Connection conn = DriverManager.getConnection(dbURL, username, password);
	String query = "Select * FROM Item WHERE ID = 110";
	Statement stmt = conn.createStatement();
	ResultSet rs = stmt.executeQuery(query);
	while (rs.next()) {
		System.out.println("ID:" + rs.getInt("Id"));
		System.out.println("Description:" + rs.getString("Descrip"));
		System.out.println("Price:" + rs.getDouble("Price"));
		System.out.println("Quantity:" + rs.getInt("Quantity"));
	}
} catch (SQLException se) {
	System.out.println("Error");
}
```
Assume that:
The required database driver is configured in the classpath.
The appropriate database is accessible with the dbURL, userName, and passWord exists.
The SQL query is valid.

What is the result?

A
An exception is thrown at runtime.

B

Compilation fails.

C
The code prints Error.

D(ans)
The code prints information about Item 110.

題解
ResultSet 物件的 getX 方法所傳入的欄位名稱參數是不區分大小寫的，所以如果 ID = 110 的 Item 資料列存在的話，就會輸出該列的資訊。

(79)
```
public class Foo<K, V> {

	private K key;
	private V value;

	public Foo(K key, V value) {
		this.key = key;
		this.value = value;
	}

	public static <T> Foo<T, T> twice(T value) {
		return new Foo<T, T>(value, value);
	}

	public K getKey() {
		return key;
	}

	public V getValue() {
		return value;
	}
}
```

Which option fails?

A
```
Foo<String, Integer> mark = new Foo<String, Integer> ("Steve", 100);
```

B
```
Foo<String, String> pair = Foo.<String>twice ("Hello World!");
```

C(ans)
```
Foo<?, ?> percentage = new Foo<?, ?>(97, 32);
```

D
```
Foo<String, String> grade = new Foo<>("John", "A");
```

題解
選項 A，雖然是正確的泛型使用方式，但是在 Java 7 以後的版本，在實體化物件的時候，最好是省略 new 運算子泛型的型態，變成以下這樣：
```
Foo<String, Integer> mark = new Foo<> ("Steve", 100);
```

選項 B，正確的泛型使用方式。
選項 C，錯誤的泛型使用方式，「?」代表未知型態。應改成以下這樣：
Foo<?, ?> percentage = new Foo<>(97, 32);
或是指定明確的類別型態：
Foo<?, ?> percentage = new Foo<Integer, Integer>(97, 32);
選項 D，正確的泛型使用方式。

(80)
Given that /green.txt and /colors/yellow.txt are accessible, and the code fragment:
Path source = Paths.get("/green.txt");
Path target = Paths.get("/colors/yellow.txt");
Files.move(source, target, StandardCopyOption.ATOMIC_MOVE);
Files.delete(source);

Which statement is true?

A
The green.txt file content is replaced by the yellow.txt file content and the yellow.txt file is deleted.

B
The yellow.txt file content is replaced by the green.txt file content and an exception is thrown.

C
The file green.txt is moved to the /colors directory.

D(ans)
A FileAlreadyExistsException is thrown at runtime.

題解
程式第 9 行，會將「/green.txt」移動到「/colors/yellow.txt」，原先的「/colors/yellow.txt」會被覆蓋，而「/green.txt」會被刪除。這裡使用了 StandardCopyOption.ATOMIC_MOVE 這個 CopyOption，會將移動檔案的動作變成是 atomic 的，也就是在這個過程中並不會被其他的執行緒介入(像是搶奪檔案的寫入權限)，。
選項 A，寫反了。
選項 B，由於「/colors/yellow.txt」已經存在，也沒使用 StandardCopyOption.REPLACE_EXISTING 這個 CopyOption，因此會在移動檔案之前拋出 FileAlreadyExistsException。檔案並不會被移動。
選項 C，並不會。
選項 D，會拋出 FileAlreadyExistsException，原因見選項 B。
注意這邊的程式實際執行時可能會有一些問題，算是 Java 的 BUG。筆者在 Linux 作業系統上中的 EXT4 檔案系統執行這段程式時，StandardCopyOption.ATOMIC_MOVE 這個 CopyOption 會強制覆蓋已存在的檔案，因此不會拋出 FileAlreadyExistsException。當程式執行到第 10 行時會拋出 NoSuchFileException，因為來源檔案在搬移之後就會被刪除。

(81)
The data.doc, data.txt and data.xml files are accessible and contain text.
Given the code fragment:
Stream<Path> paths = Stream.of(Paths.get("data.doc"),
Paths.get("data.txt"),
Paths.get("data.xml"));

```
paths.filter(s -> s.toString().endsWith("txt")).forEach(
            s -> {
                  try {
                        Files.readAllLines(s)
                        .stream()
                        .forEach(System.out::println); //line n1
                  } catch (IOException e) {
                        System.out.println("Exception");
                  }
            });
```
What is the result?

A(ans)
The program prints the content of data.txt file.

B
The program prints:
Exception
〈 The content of the data.txt file 〉
Exception

C
A compilation error occurs at line n1.

D
The program prints the content of the three files.

題解
程式第 24 行開始會保留路徑為「txt」結尾的 Path 物件，接著讀取剩下來的路徑所指到的檔案之所有的內容，並輸出至標準輸出串流中。paths 變數參考到的串流物件中的 Path 物件，只有「data.txt」會被 filter 保留，因此只會讀取並輸出「data.txt」的檔案內容。

(82)
```
class Book {

      int id;
      String name;

      public Book(int id, String name) {
            this.id = id;
            this.name = name;
      }

      public boolean equals(Object obj) { //line n1
            boolean output = false;
            Book b = (Book) obj;
            if (this.name.equals(b.name)) {
                  output = true;
            }
            return output;
      }
}
```
and the code fragment:
```
Book b1 = new Book(101, "Java Programing");
```

```
Book b2 = new Book(102, "Java Programing");
System.out.println(b1.equals(b2)); //line n2
```

Which statement is true?

A(ans)
The program prints true.

B
The program prints false.

C
A compilation error occurs. To ensure successful compilation, replace line n1 with:
boolean equals (Book obj) {

D
A compilation error occurs. To ensure successful compilation, replace line n2 with:
System.out.println (b1.equals((Object)b2));

題解
Book 類別覆寫了 equals 方法，當兩個 Book 物件的 name 欄位邏輯相同時，則這兩個物件也符合邏輯上的相同。
所以 b1 和 b2 變數所參考到的 Book 物件，有邏輯上的相同，會輸出「true」。

(83)

```
interface CourseFilter extends Predicate<String> {

	public default boolean test(String str) {
		return str.equals("Java");
	}
}
```

And

```
List<String> strs = Arrays.asList("Java", "Java EE", "Java ME");
Predicate<String> cf1 = s -> s.length() > 3;
Predicate cf2 = new CourseFilter() { //line n1
	public boolean test(String s) {
		return s.contains("Java");
	}
};
long c = strs.stream()
		.filter(cf1)
		.filter(cf2) //line n2
		.count();
System.out.println(c);
```

What is the result?

A
2

B(ans)
3

C

A compilation error occurs at line n1.

D
A compilation error occurs at line n2.

題解
CourseFilter 介面繼承了 Predicate 介面，並使用 Java 8 之後新出的預設方法功能來實作 Predicate 介面的 test 方法，判斷傳入 test 方法的參數是否為「Java」字串。
程式第 16 行，使用 Java 8 之後新出的 Lambda 語法來快速實作出 Predicate 介面的 test 方法，判斷傳入 test 方法的參數所參考的字串物件的長度是否大於 3。
程式第 17 行，使用匿名類別實作 CourseFilter 介面，再次覆寫已經在 CourseFilter 介面中用預設方法實作過的 test 方法，判斷傳入 test 方法的參數是否包含「Java」子字串。
程式第 22 行，將集合轉為串流。
程式第 23 行，保留長度大於 3 的字串物件，其餘的丟棄。在此所有元素都會被保留。
程式第 24 行，保留集合中的包含「Java」子字串的字串，其餘的丟棄。在此所有元素都會被保留。
程式第 25 行，計算集合的元素數量。此時元素數量為 3。
程式第 26 行，輸出集合的元素數量「3」。

(84)
```
class Vehicle {

	int distance;

	Vehicle(int x) {
		this.distance = x;
	}

	public void increSpeed(int time) {
		int timeTravel = time;
		class Car { // line n1

			int value = 0;

			public void speed() {
				value = distance / timeTravel; // line n2
				System.out.println("Velocity with new speed " + value + "
kmph");
			}
		}
		new Car().speed(); // line n3
	}
}
```
and this code fragment:
```
Vehicle v = new Vehicle(100);
v.increSpeed(60);
```

A(ans)
Velocity with new speed 1 kmph

B
A compilation error occurs at line n1.

C
A compilation error occurs at line n2.

D
A compilation error occurs at line n3.

題解
Java 8 之後的版本，允許方法內的區域內部類別(local inner class)，使用方法內的區域變數而不一定要是常數，但僅能取得變數數值，無法更改變數數值。
在這個題目中，Vehicle 類別和其方法內的區域內部類別 Car，都可以正常的編譯。程式實體化出 Vehicle 物件之後，會透過建構子將 distance 物件欄位設為 100。接著使用 Vehicle 物件的 increSpeed 方法，呼叫其 Car 區域內部類別物件中的 speed 方法，設定 value 變數的值為 distance 欄位的值 100，再除上傳入 increSpeed 方法的參數值 60。由於變數型態均為整數，因此進行除法運算時，會自動捨棄掉小數點的部份，value 變數的值會被重新指派為 1。程式第 17 行，將 value 的值輸出。

(85)
```
public class MyFor3 {

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

What is the result?

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。

(86)
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」。

(87)
```
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

(88)
```
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 」。

(89)
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」。

(90)
```
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。

(91)

```
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 MissingInfoException, 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。

(92)
```
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。

(93)
```
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(ams)
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。

(94)
```
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。

(95)
A method is declared to take three arguments. A program calls this method and passes only two arguments. What is the results?

A(ans)
Compilation 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。

(96)
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。

(97)

```
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。

(98)

```
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.

E
message =

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 everyone! -> Hi XvXryonX!
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。

(99)
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。

(100)

```
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。

(101)

```
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。

(102)

```
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。

(103)

```
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。

(104)
```
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!」。

(105)
```
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?

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」。

(106)
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。

(107)
```
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。

(108)
```
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」。

(109)

```
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。

(110)
```
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

(111)

```
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
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。

(112)

```
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)
4

B(ams)
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。

(113)

```
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

(114)

```
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 0

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);
            }
      }
}
```

(115)

```
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

E
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：

(116)

```
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);
            }
      }
}
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(nas)
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);
		}
	}
}
```

(117)

```
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) {
}

(118)
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
D. 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。

(119)

```
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(ams)
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。

(120)
```
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「...」。

(121)

```
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。

(122)

```
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。

(123)
```
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。

(124)

```
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。

(125)
```
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

(126)
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：
Text.java
```
package org.magiclen;

import java.util.Scanner;
import java.util.Base64;

public class Test {
```

```
	...
}
```
Text.java
```
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());
	}
}
```

(127)
```
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。

(128)
```
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。