



62

Questions in Java

HammooD

Damascus university 2010

Special thanks to :

magician

Mu_Nizar

Mohammad_807

1 – إن نتيجة تنفيذ البرنامج التالي هي :

```
class Father {  
    public static void Question1(){  
        System.out.println("I am the Method in Father Class");  
    }  
}  
  
class Son extends Father {  
    public void Question1(){  
        System.out.println("I am the Method in Son Class");  
    }  
}  
  
public class Main {  
    public static void main(String[] args) {  
        Son MySon = new Son();  
        MySon.Question1();  
    }  
}
```

1 – طباعة الجملة I am the Method in Son Class

2 – طباعة الجملة I am the Method in Father Class

3 – يوجد خطأ Compiler لأنه لا يمكن القيام بـ override للتابع Question1 لأنه static .

4 – حدوث خطأ Compiler لأن التابع Question1 في الصف Father لا يمكن أن يكون من النوع static .

2 – بتعديل البرنامج السابق ليصبح على الشكل التالي تكون نتيجة التنفيذ هي :

```
class Father {  
    public static void Question1(){
```

```

        System.out.println("I am the Method in Father Class");
    }
}

class Son extends Father {

    public static void Question1(){

        System.out.println("I am the Method in Son Class");

    }

}

public class Main {

    public static void main(String[] args) {

        Father MySon = new Son();

        MySon.Question1();

    }

}

```

1 – طباعة الجملة I am the Method in Son Class

2 – طباعة الجملة I am the Method in Father Class

3 – يوجد خطأ Compiler لأنه لا يمكن القيام بـ override للتابع Question1 لأنه static .

4 – حدوث خطأ Compiler لأن التابع Question1 في الصف Father لا يمكن أن يكون من النوع static .

3 – نجعل التابع Question1 غير static فتصبح نتيجة التنفيذ للبرنامج التالي هي :

```

class Father {

    public void Question1(){

        System.out.println("I am the Method in Father Class");

    }

}

```

```

class Son extends Father {

    public void Question1(){

        System.out.println("I am the Method in Son Class");

    }

}

public class Main {

    public static void main(String[] args) {

        Father MySon = new Son();

        MySon.Question1();

    }

}

```

1 – طباعة الجملة I am the Method in Son Class

2 – طباعة الجملة I am the Method in Father Class

3 – حدوث حلقة لا نهائية من الطباعة و حصول StackOverFlow Exception .

4 – يوجد خطأ Compiler في تعريف MySon في التابع main (MySon can't be declared as an instance from Son Class).

4- نتيجة تنفيذ البرنامج التالي هي :

```

static class Father {

    public void Question1(){

        System.out.println("I am the Method in Father Class");

    }

}

class Son extends Father {

    public void Question1(){

```

```

        System.out.println("I am the Method in Son Class");
    }
}

public class Main {

    public static void main(String[] args) {

        Father MySon = new Son();

        MySon.Question1();
    }
}

```

1 – طباعة الجملة I am the Method in Son Class

2 – طباعة الجملة I am the Method in Father Class

3 – يوجد خطأ Compiler لأنه لا يمكن القيام بـ override للتابع Question1 لأنه الصف الذي يحويه هو صف static .

4 – حدوث خطأ Compiler لأن الصف Father هو صف static .

5 - إن محاولة ترجمة الكود التالي هي :

```

class MyClass {

    int i = 5;

    static {

        int i;

    }

}

```

1 – يوجد خطأ Compiler لأن المتحول i تم تعريفه أكثر من مرة (i is already defined in Class quiz.MyClass).

2 – يوجد خطأ Compiler لأن المتحول i يجب تهيئته بقيمة داخل الـ Scope الخاصة بالمتحولات الـ static .

3 – الكود السابق خاطئ و كان من الممكن أن يكون صحيحاً لو أن المتحول i (الأول) كان من النوع static .

4 – الكود السابق صحيح و لا يحوي أية أخطاء.

6 – إن محاولة ترجمة الجزء التالي من أحد البرامج هي :

```
class Father {  
  
    int i;  
  
    Father(int i)  
  
    {  
  
        System.out.println("I am Father Constructor");  
  
    }  
}  
  
class Son extends Father {  
  
}
```

1 – الترجمة ستكون ناجحة و لا توجد فيها أية أخطاء.

2 – سينتج خطأ Compiler لأنه يجب كتابة الباني الافتراضي للصف Son من قبل المبرمج.

3 – سينتج خطأ Compiler لأنه يجب استدعاء الباني الخاص بالصف Father الذي قمنا بكتابته في أحد بواني الصف Son .

4 – الكود السابق يتسبب في Runtime Exception بسبب الغاء عمل الباني الافتراضي في الصف Father بعد أن قمنا بكتابة باني غير الباني الافتراضي .

5 – الإجابة ليست أي من الإجابات السابقة .

7 – لدينا الصف MyClass الذي يحوي التابع التالي :

```
class MyClass {  
  
    int MyClass()  
  
    {  
  
        return 3;  
  
    }  
}
```

```
}  
  
}
```

- 1 – إن التابع السابق هو باني خاص بالتابع MyClass .
- 2 – التابع السابق هو تابع عادي و لا يعتبر بانياً .
- 3 – يوجد خطأ Compiler في التابع السابق لأنه لا يمكن أن يرد قيمة و يكون من نفس اسم الصف .
- 4 – يمكن تجاوز الخطأ الموجود في الخيار السابق عن طريق جعل التابع السابق static .
- 5 – الإجابة الصحيحة مختلفة عن ما سبق .

8 - إن الخطأ الذي سيظهر في الكود التالي هو :

```
class MyClass  
{  
    static MyClass()  
    {  
        System.out.println("I am The First Statement here!");  
        this();  
    }  
}
```

- 1 – سيظهر خطأ وحيد هو أن الباني لا يمكن أن يكون static .
- 2 – سيظهر خطأ وحيد أنه لا يمكن أن نكتب تعليمة this داخل التتابع الـ static .
- 3 – سيظهر خطأ وحيد أن تعليمة this يجب أن تكون أول تعليمة في الباني .
- 4 – سيظهر الخطأ الأول و الثالث فقط .
- 5 – ستظهر الأخطاء الثلاثة الأولى جميعها .

9 – إن محاولة تنفيذ البرنامج التالي هي :

```
public class Main {  
    Main obj = new Main();  
}
```



```

void print(){

    System.out.println("inside print Method");

}

public static void main(String[] args) {

    this.print();

}

}

```

- 1 – سيتم التنفيذ بنجاح و تتم طباعة العبارة inside print Method .
- 2 – يوجد خطأ في تعريف المتحول obj .
- 3 – يوجد خطأ داخل التابع main لأنه لا يمكن استعمال التعليمة this داخله .
- 4 – الإجابة 2 و 3 .
- 5 – الإجابة الصحيحة مختلفة عما سبق .

10 – إن محاولة تنفيذ البرنامج التالي هي :

```

public class Main {

    static int i;

    public static void main(String[] args) {

        int y;

        i = y;

        System.out.println(i);

    }

}

```

- 1 – يوجد خطأ Compiler في تعريف المتحول y لأنه لا يمكن تعريفه داخل تابع static .
- 2 – ستتم طباعة قيمة i و ستكون 0 .
- 3 – ستتم طباعة قيمة i و ستكون قيمة عشوائية متغيرة عند كل تنفيذ .

4 – يوجد خطأ Compiler لأنه يجب تهيئة المتحول y قبل استخدامه في التابع main .

5 – الأجوبة السابقة جميعها خاطئة لأنه سيحدث NullPointerException بسبب عدم تهيئة المتحول y قبل استخدامه .

11 – إن نتيجة تنفيذ البرنامج التالي هي :

```
public class Main {  
    String[] str;  
  
    public void I_Will_Call_main()  
    {  
        main(str);  
    }  
  
    public static void main(String[] args) {  
        I_Will_Call_main();  
    }  
}
```

1 – ستحصل استدعاءات غير منتهية أثناء التنفيذ بين كل من التابعين السابقين لأن كل منهما يستدعي الآخر .

2 – يوجد خطأ Compiler لأنه لا يمكن أن يتم استدعاء التابع main الذي هو static من قبل تابع آخر ليس static .

3 – يوجد خطأ Compiler لأنه لا يمكن أن يتم استدعاء التابع I_Will_Call_main الذي هو غير static من قبل التابع main الذي يحمل الصفة static .

4 – سنحصل على الخطأين الموجودين في الخيار الثاني و الثالث .

5 – الأجوبة السابقة جميعها خاطئة لأن الـ Compiler سيتجاهل استدعاء التابع main في أي تابع آخر و يتابع تنفيذ تعليمات البرنامج بشكل طبيعي .

12 – إن محاولة ترجمة الكود التالي ستكون :

```
public class Main {  
  
    public String toString()  

```

```

{
    return "String"+ this;
}

public static void main(String[] args) {

    Main obj = new Main();

    obj.toString();

}
}

```

- 1 – طباعة العبارة String عدد لا نهائي من المرات دون توقف.
- 2 – طباعة العبارة String و اسم الكائن (obj) عدد لا نهائي من المرات دون توقف.
- 3 – طباعة العبارة String و اسم الكائن (obj) مرة واحدة فقط .
- 4 – طباعة اسم الكائن ثم العبارة String ثم اسم الكائن مرة أخرى وذلك لمرة واحدة فقط .
- 5 – جميع ما سبق خاطئ لأننا سنحصل على StackOverflowError .

13 – ليكن لدينا الكود التالي :

```

interface Monster {

    void menace();

}

interface DangerousMonster extends Monster {

    void destroy();

}

abstract class DragonZilla implements DangerousMonster {

    public void menace() {}

}

```

1 – سنحصل على خطأ Compiler لأن الصف المجرد (DragonZilla) لا يمكن أن يقوم بـ implement لـ interface .

2 – سنحصل على خطأ Compiler لأن الـ interface (DangerousMonster) لا يمكن أن تقوم بعملية implements لـ interface أخرى .

3 – سنحصل على خطأ Compiler لأن الصف DragonZilla لم يتم بإعادة تعريف التابع destroy .

4 – يوجد خطأ في الكود السابق و كان من الممكن أن يكون صحيحاً لو أن الصف DragonZilla لم يكن abstract .

5 – جميع الأجوبة السابقة خاطئة لأن الكود السابق صحيح و لا يحوي أية أخطاء .

14 – نضيف صفّاً إضافياً للكود السابق ليصبح كما يلي :

```
interface Monster {  
    void menace();  
}  
  
interface DangerousMonster extends Monster {  
    void destroy();  
}  
  
abstract class DragonZilla implements DangerousMonster {  
    public void menace() {}  
}  
  
class MyClass extends DragonZilla {  
}
```

1 – سنحصل على خطأ Compiler لأن الصف المجرد (DragonZilla) لا يمكن أن يقوم بـ implement لـ interface .

2 – سنحصل على خطأ Compiler لأن الـ interface (DangerousMonster) لا يمكن أن تقوم بعملية implements لـ interface أخرى .

3 – سنحصل على خطأ Compiler لأن الصف MyClass لم يتم بإعادة تعريف التابع destroy .

4 – يوجد خطأ في الكود السابق و كان من الممكن أن يكون صحيحاً لو أن الصف MyClass كان abstract .

5 – الخياران 3 و 4 .

15- إن المتحولات في أي interface هي final و static :

1 – العبارة السابقة صحيحة.

2 – العبارة السابقة خاطئة.

16 – إن كتابة متحولات private في أي interface سيتسبب بـ :

1 – Compiler Error

2- RuntimeException

3 – Warning

4 – Exception و لكن ليس من النوع RuntimeException

5 – يمكن وضع متحولات private في interface دون أن يتسبب ذلك بأي خطأ .

17 – أي من الأسطر البرمجية التالية صحيح :

- 1- abstract void function(){};
- 2- void function();
- 3- abstract void function(int y);
- 4- int void function(int y){};
- 5- none of the above.

18 – أي من الأسطر البرمجية التالية صحيح :

- 1- static interface Myinterface {}
- 2- private interface Myinterface {}
- 3- protected interface Myinterface {}
- 4- final interface Myinterface {}
- 5- none of the above.

19 – أي من الأسطر البرمجية التالية خاطئ :

- 1- interface SecondInterface extends Myinterface{}
- 2- interface SecondInterface extends Thread{}
- 3- interface SecondInterface implements Myinterface{}

- 4- interface SecondInterface extends Myinterface extends Thread{}
- 5- 2 & 3 & 4.

20 – إن محاولة تنفيذ البرنامج التالي هي :

```
class Father {  
  
    void MethodToOverride(){  
  
        System.out.println("I am The Method in Father Class");  
  
    }  
}  
  
class Son extends Father {  
  
    private void MethodToOverride(){  
  
        System.out.println("I am The Method in Son Class");  
  
    }  
}  
  
public class Main {  
  
    public static void main(String[] args) {  
  
        Father son = new Son();  
  
        son.MethodToOverride();  
  
    }  
}
```

1 – طباعة العبارة I am The Method in Son Class .

2- يوجد خطأ Compiler لأن التابع MethodToOverride هو تابع private و لا يمكن استدعائه من الكائن son الذي أنشأناه في الصف Main .

3 – يوجد خطأ Compiler في الـ override للتابع MethodToOverride عند الابن حيث لا يمكن أن يكون private .

4 – طباعة العبارة I am The Method in Father Class .

5 – كان من الممكن أن يكون البرنامج قابلاً للتنفيذ لو أن التابع MethodToOverride عند الابن كان static ولم يكن private.

6 – الإجابتان 3 و 5 صحيحتان .

21 – بتغيير طفيف على البرنامج السابق تصبح نتيجة التنفيذ هي :

```
class Father {  
  
    void MethodToOverride(){  
  
        System.out.println("I am The Method in Father Class");  
  
    }  
}  
  
class Son extends Father {  
  
    public void MethodToOverride(){  
  
        System.out.println("I am The Method in Son Class");  
  
    }  
}  
  
public class Main {  
  
    public static void main(String[] args) {  
  
        Father son = new Son();  
  
        son.MethodToOverride();  
  
    }  
}
```

1 – طباعة العبارة I am The Method in Son Class .

2 – يوجد خطأ Compiler في الـ override للتابع MethodToOverride عند الابن حيث لا يمكن أن يكون public دون أن يكون public عند الأب أيضاً .

3 – طباعة العبارة I am The Method in Father Class .

4 – كان من الممكن أن يكون البرنامج قابلاً للتنفيذ لو أن التابع MethodToOverride عند الابن كان static ولم يكن public .

5 – الإجابتان 2 و 4 صحيحتان .

22 - أي من الإسنادات التالية خاطئ ؟

```
class MyClass{}

class A extends MyClass {

    void doit(){};

}

class B extends A{

    void dothisa(){};

}

class C{ }

public class Main {

    public static void main(String[] args) {

        MyClass ss=new MyClass();

        A a=new A();

        B b=new B();

        C c=new C();

        b.dothisa();

        a=b;

        a.doit();

        a=(A)ss;//1

        a.doit();

        b=(B)a;//2
```



```

        b.dothisa();

        b=(B)c;//3
    }
}

```

- 1 - الإسناد في السطر 3
- 2 - الإسناد في السطرين 2-3
- 3 - الإسناد في الأسطر 3-2-1
- 4 - الإسناد في السطرين 2-1
- 5 - الإسنادات السابقة كلها صحيحة .

23 – إن نتيجة تنفيذ الكود التالي هي :

```

class outer{
    private int x;
    void nonStaticMethod(){
    public static void dothis()
    {
        class inner_inside_method
        {
            void doit()
            {
                x=1;
                nonStaticMethod(); //1
            }
        }
    }
}
}

```

- 1 - يعطي compiler error لأن مكان تعريف الكلاس inner_inside_method خاطئ.
- 2 - سنحصل على compiler error لأنه لا يمكن كتابة inner class في تابع يحمل الصفة static .
- 3 - كان من الممكن أن نتجاوز الخطأ الموجود في الخيار 2 لو أن الـ inner class كان من النوع static .
- 4 - سنحصل على compiler error عند استدعاء التابع nonStaticMethod لأنه تم وضع تابع غير static داخل محتوى static (static context).
- 5 - الإجابة 2 و 3 .
- 6 - الكود صحيح تماماً

```
class outer{
    private int x;

    public void dothis(){
    }

    class inner_inside_method
    {
    }
}
```

لاحظ أنه في الكود السابق يمكن أن يكون الصف inner_inside_method :

- public - 1
- static - 2
- protected - 3
- private - 4

24 – إن نتيجة تنفيذ الكود السابق هي :

```
class myint{int x;}

public class Main {
    static myint x;

    int y;

    static{
        x=new myint();

        x.x=1;//1

        System.out.print("x");//2
    }
}
```

```

}

{y=1;} //3

public static void main(String[] args ){

    System.out.println(x.x);

}

}

```

1 - compiler error لوجود السطر 1

2 - x1

3 - compiler error لوجود السطر 2

4 - compiler error لوجود السطر 3

5 - البرنامج السابق يعطي RuntimeException أثناء التشغيل.

25 - ما نتيجة تنفيذ الكود التالي باعتبار أننا قمنا بـ `import java.util.Stack` :

```

class myClass{

    Stack s=new Stack();

    void myMethod(){

        s.push("this");

        s.push("is");

        s.push("test");

        String s2 = "";

        s2=s2+s.pop();//1

        s2=s.pop()+s.pop();//2

    }

}

```

1 - Compiler Error في 1

2 - Compiler Error في 2

3 - Compiler Error في 1 و 2

4 - الكود السابق صحيح تماماً .

26 - لدينا البرنامج التالي :

```
public class Main {  
    public static void main(String[] args ){  
        try  
        {  
            System.out.print(" nothing to try");  
        }  
        catch(something e)  
        {  
            e.toString();  
        }  
    }  
}
```

إن something ممكن أن تكون :

- 1 - Throwable& Exception
- 2 - FileNotFoundException& IOException
- 3 - Exception& FileNotFoundException
- 4 - FileNotFoundException& Throwable
- 5 - جميع الخيارات السابقة خاطئة.

27 - أي من الأسطر في البرنامج التالي سيعطي Compiler Error أو Exception :

```
public class Main {  
    public static void main(String[] args ){  
        Integer i1=null;//1  
        int i2=null;//2  
        Integer i3=new Integer(null);}//3
```

```
}
```

- .2 Compiler error - 1
- .3 Exception - 2
- .1 Exception - 3
- .2 Compiler error,3 Exception - 4
- .2 Exception - 5

28 – إن نتيجة تنفيذ الكود التالي هي :

```
public class Main {  
    void show(){System.out.println("test");}  
    String s;  
    public static void main(String[] args ){  
        show();//1  
        s="test" ;//2  
    }  
}
```

- 1 - طباعة العبارة test.
- 2 - Compiler Error بسبب السطر 1.
- 3 - Compiler Error بسبب السطر 2.
- 4 - Compiler Error بسبب السطرين 1 و 2.
- 5 - سيحصل Exception أثناء التشغيل لأننا لم نقوم بعملية new للمتحول s.

29 – أي من الأسطر التالية سيظهر عنده Compile Error :

```
class MyClass {  
    int i1;  
    Integer i2 = new Integer(5);  
}  
  
public class Main {  
    int i3= 8;  
    public static void main(String[] args) {
```

```

Integer i4 = new Integer(null);
Integer i5= new Integer(null);
String s = new String();
s = i5;//5

System.out.println(new MyClass().i1);//1
System.out.println(new MyClass().i2);//2
System.out.println(i3);//3
System.out.println(i4);//4
}
}

```

1,2 - 1
 3,5 - 2
 5 - 3
 1,5 - 4
 4,5,3 - 5

30 – نتيجة تنفيذ الكود التالي هي :

```

public class Main {
    public static void main(String[] args) {
        String SS = new String();
        System.out.println(SS);
    }
}

```

- 1 - طباعة سطر فارغ.
- 2 - Compiler Error عند تعليمة الطباعة .
- 3 - Exception عند تعليمة الطباعة.
- 4 - طباعة قيمة عشوائية مختلفة عند كل تنفيذ.

31 – لكي يتم طباعة العبارة do something في الكود التالي يجب أن تكون قيمة equals هي :

```
public class Main {  
    public static void main(String[] args) {  
        String str1=new String("test1");  
        String str2=new String("test1");  
        String str3="test1";  
        String str4="test1";  
        if( equals )  
            {System.out.println("do something");}  
    }  
}
```

str1==str2 - 1

str1==str3 - 2

str1==str4 - 3

str3==str4 - 4

5 - إن كل القيم السابقة صحيحة.

32 – لكي لا يتم طباعة العبارة do something في الكود التالي يجب أن تكون notEquals هي :

```
public class Main {  
    public static void main(String[] args) {  
        String str1=new String("test1");  
        String str2=new String("test1");  
        String str3="test1";  
        String str4="test1";  
        if( notEquals )  
            {System.out.println("do something");}
```

```
}  
}
```

.str1.equals(str2) - 1
.str1.equals(str3) - 2
.str3.equals(str4) - 3
4 - جميع الأجوبة السابقة خاطئة.
5 - جميع الأجوبة السابقة صحيحة.

33 – إن نتيجة تنفيذ الكود التالي هي :

```
class Ops {  
    int i,j;  
}  
  
public class Main {  
    public static void main(String[] args) {  
        Integer n1 = new Integer(47);  
        Integer n2 = new Integer(47);  
        System.out.print (n1 == n2);  
        System.out.print (n1 != n2);  
        System.out.print (n1 .equals(n2));  
        Ops obj1 = new Ops();  
        Ops obj2 = new Ops();  
        obj2.i=obj2.j=1;  
        System.out.print (obj1.equals(obj2));  
    }  
}
```

.true false true true - 1
.true false false false - 2
.false true true false - 3
.false true false true - 4

5 - false true false false.

34 – إن نتيجة تنفيذ البرنامج التالي هي :

```
class Test {  
    void Test () {  
        System.out.print("First ");  
        return;  
    }  
    Test(){  
        System.out.print("Second ");  
    }  
}
```

```
public class Main {  
    public static void main(String[] args) {  
        Test T = new Test();  
        T.Test();//1  
    }  
}
```

1 - First Second.

2 - Second First.

3 - يوجد Compiler Error بسبب تشابه اسمي التابعين في الصف Test .

4 - سيحصل Exception أثناء التشغيل .

5 - يوجد Compiler Error عند استدعاء التابع Test في الـ main (السطر 1).

35 – إن نتيجة تنفيذ البرنامج التالي هي :

```
class Overloading{  
    Overloading(int i){
```

```

        System.out.print("I am int Constructor");
    }
    Overloading(Integer i){
        System.out.print("I am Integer Constructor");
    }
    Overloading(float f){
        System.out.print("I am Constructor with f ");
    }
    Overloading(){
        System.out.print("I am default Constructor ");
    }
}

public class Main {
    public static void main(String[] args) {
        Overloading O = new Overloading(5);//1
    }
}

```

.I am int Constructor - 1
 .I am Integer Constructor - 2
 .I am default Constructor I am int Constructor - 3
 .I am default Constructor I am Integer Constructor - 4
 .I am Constructor with f - 5

36 – نستبدل السطر 1 في البرنامج السابق بالسطر `new Overloading(5.5);//1` فتصبح
 نتيجة التنفيذ :

I am Constructor with f - 1
 I am default Constructor - 2
 .I am Integer Constructor - 3
 .I am int Constructor - 4
 compiler error - 5

37 – نستبدل السطر 1 في البرنامج السابق بالسطر 1//new Overloading(f); فتصبح نتيجة التنفيذ :

- 1 - I am Constructor with f
- 2 - سيتم التنفيذ و لكن لا يوجد خرج.
- 3 - Compiler Error .
- 4 - يوجد Compiler Error و نستطيع تصحيحه عن طريق إضافة بانٍ آخر يأخذ القيمة String في الصف Overloading .
- 5 - الأجوبة السابقة جميعها خاطئة لأن ما سيحصل هو RuntimeException .

38 – نعدل البرنامج السابق ليصبح كما يلي فتكون نتيجة التنفيذ هي :

```
class Overloading{
    Overloading(int i){
        System.out.print("I am int Constructor");
    }
    Overloading(Integer i){
        System.out.print("I am Integer Constructor");
    }
    Overloading(double d){
        System.out.print("I am Constructor with d ");
    }
    Overloading(String s){
        System.out.print("I am String Constructor ");
    }
}

public class Main {
    public static void main(String[] args) {
        Overloading O = new Overloading("5".hashCode());
    }
}
```

```

    }
}

```

I am Integer Constructor - 1

I am String Constructor - 2

I am int Constructor - 3

Compiler Error - 4 بسبب عدم القيام بعملية new للقيمة "5" التي تم تمريرها للتابع في الـ . main

Compiler Error - 5 بسبب التضارب بين الباني Overloading الذي يأخذ وسيطاً من النوع int و الباني الآخر الذي يأخذ وسيطاً من النوع String .

39 – الكود التالي يمثل جزءاً من أحد البرامج فهل هو صحيح أم يحوي Compiler Errors ؟:

```

class OverLL {

    public int OverLL(int i) {

        System.out.println("int Value");

        return 0;

    }

    public String OverLL(int i) {

        System.out.println("String Value");

        return new Integer(4).toString();

    }

}

```

1 - الكود السابق صحيح.

2 - يحوي Compiler Error وحيد.

3 - يحوي 2 Compiler Errors.

40 – هل يوجد خطأ في البرنامج التالي أم لا ؟ :

```

public class Main {

    public static void main(String[] args) {

        throw new ClassNotFoundException();//1

    }

}

```

}

- 1 - لا يوجد أية أخطاء و البرنامج صحيح و قابل للتنفيذ.
- 2 - يوجد Compiler Error و من ممكن تصحيحه عن طريق إضافة try, catch لتعليمة throw .
- 3 - يوجد Compiler Error و من الممكن تصحيحه عن طريق إضافة throws
- 4 - يوجد Compiler Error و كان من الممكن عدم وجود هذا الخطأ لو أن الـ ClassNotFoundException للتابع main .
- 5 - الإجابتان 2 و 3 صحيحتان.
- 6 - الأجوبة 2 و 3 و 4 صحيحة.

41 – ما ناتج تنفيذ البرنامج التالي :

```
public class Main {  
    public static void main(String[] args) {  
        Integer[] array = new Integer[2];  
        int x=0 ;  
        array[x]=x=1;//1  
  
        System.out.println(array[0]);  
        System.out.println(array[1]);//2  
    }  
}
```

- 1 - طباعة 1 ثم طباعة null.
- 2 - طباعة null ثم طباعة 1 .
- 3 - طباعة null ثم طباعة null .
- 4 - يوجد Compiler Error عند السطر 1 .
- 5 - سيحصل NullPointerException عند السطر 2 .

42 – لدينا الصفوف الثلاثة التالية :

```
class a  
{
```

```

void eat (){}

public a(){
    System.out.println(" in a");
    c cObject = new c(8);
    eat();
}
}

class c {
    b bObject = new b();
    c(int i) {
        System.out.println(" in c");
        System.out.println(i);
    }
}

class b extends a
{
    private Integer color = new Integer(1);
    void eat ()
    {System.out.println(color);}

    void b()
    {System.out.println(" in b");}
}

```

ما هو ناتج تنفيذ الكود التالي على الصفوف الثلاثة السابقة ؟ :

```

public class Main {
    public static void main(String[] args) {
        b ObjectInb = new b();
    }
}

```

```
}  
  
}
```

- 1 - طباعة الجملة `b in` لمرة واحدة فقط .
- 2 - طباعة الجملة `c in` لوحدها عدد لا نهائي من المرات دون توقف.
- 3 - طباعة الجمل `a in` و `c in` و `b in` على التناوب و ذلك لعدد لا نهائي من المرات.
- 4 - طباعة الجملة `a in` لوحدها عدد لا نهائي من المرات دون توقف.
- 5 - حدوث `StackOverflowError` بعد فترة قصيرة من تشغيل البرنامج و توقف البرنامج عن العمل.

43 – إن نتيجة تنفيذ البرنامج التالي هي :

```
public class DirtyClass {  
    public static void main(String[] args) {  
        for (int i = 0; i < 5; i++) {  
            System.out.println("Simple Text");//1  
            System.err.println("Error!");//2  
        }  
    }  
}
```

- 1 - طباعة الجملة الأولى (السطر 1) خمس مرات ثم طباعة الجملة الثانية (السطر 2) خمس مرات.
- 2 - طباعة كل من الجملتين خمس مرات بالتناوب و ذلك ابتداء من الجملة الأولى .
- 3 - طباعة الجملة الأولى في بداية الخرج و الثانية في نهايته دائماً.
- 4 - لا يمكن التنبؤ بالخرج حيث سيكون مختلفاً عند كل تنفيذ.
- 5 - الأجوبة السابقة جميعها غير صحيحة.

44 – إن ناتج تنفيذ البرنامج التالي هو:

```
public class Main {  
    public void method(Object o)
```

```

{
    System.out.println("Object Version");
}

public void method(String s)
{
    System.out.println("String Version");
}

public static void main(String args[])
{
    Main question = new Main();
    question.method(null);//1
}
}

```

1 - يوجد Compiler Error عند السطر 1 .

2 - سيتم طباعة String Version .

3 - سيتم طباعة Object Version .

4 - سيحصل Exception أثناء تشغيل البرنامج بسبب السطر 1 .

45 – بعد تعديل البرنامج السابق ليصبح كالتالي تكون نتيجة التنفيذ هي :

```

public class Main {
    public void method(StringBuffer sb)
    {
        System.out.println("StringBuffer Verion");
    }

    public void method(String s)
    {
        System.out.println("String Version");
    }
}

```



```

    }

    public static void main(String args[])
    {
        Main question = new Main();
        question.method(null);//1
    }
}

```

1 - يوجد Compiler Error عند السطر 1.

2 - سيتم طباعة String Version .

3 - سيتم طباعة StringBuffer Version .

4 - سيحصل Exception أثناء تشغيل البرنامج بسبب السطر 1 .

46 – بعد تعديل البرنامج السابق ليصبح كالتالي تكون نتيجة التنفيذ هي :

```

public class Main {
    public void method(StringBuffer sb)
    {
        System.out.println("StringBuffer Verion");
    }

    public void method(String s)
    {
        System.out.println("String Version");
    }

    public static void main(String args[])
    {
        Main question = new Main();
        question.method("HammooD");//1
    }
}

```

}

- 1 - يوجد Compiler Error عند السطر 1.
- 2 - سيتم طباعة String Version .
- 3 - سيتم طباعة StringBuffer Version .
- 4 - سيحصل Exception أثناء تشغيل البرنامج بسبب السطر 1 .

47 – ليكن لدينا الـ interface التالية :

```
public interface AQuestion {  
    public abstract void someMethod() throws Exception;  
}
```

إن الصف الذي سيقوم بـ **implement** لهذه الواجهة يجب أن :

- 1 - يكون **abstract** بالضرورة.
- 2 - يملك تابعاً هو **public abstract void someMethod();**
- 3 - يملك تابعاً هو **public void someMethod()** ويجب على هذا التابع أن يقوم بـ **throws** **Exception**
- 4 - أن يملك تابعاً هو **public void someMethod()** دون ضرورة لعملية **throws Exception**.

48 - ليكن لدينا البرنامج التالي :

```
public class Main {  
    private int i = j; //1  
    private int j = 10;  
  
    public static void main(String[] args) {  
        System.out.println((new Main()).i);  
    }  
}
```

- 1 - يوجد Compiler Error لأنه لا يمكن الوصول إلى المتحول **i** من التابع **main** لأن هذا التابع **static** .

2 - يوجد Compiler Error عند السطر 1 بسبب عملية forward reference غير مسموحة.

3 - البرنامج لا يحوي أية أخطاء و الخرج هو 10 .

4 - البرنامج لا يحوي أية أخطاء و الخرج هو 0 .

5 - الخياران 1 و 2 .

49 - في الكود التالي تتغير قيمة i من 12 إلى 14 عندما :

```
class outer{  
    public static int j=12;  
    public class inner  
    {  
        inner()  
        {  
            outer.this.j=14;  
            System.out.println(outer.this.j);  
        }  
    }  
    public inner newinner() //1  
    {  
        return new inner();  
    }  
}
```

1 - يتم تعريف object من outer في الـ main.

2 - يتم تعريف object من inner في الـ main.

3 - يتم تعريف object من outer واستدعاء تابع الـ newinner() في الـ main.

4 - الجوابين 2-3.

5 - الأجوبة السابقة جميعها خاطئة.

50 – في الكود السابق طريقة تعريف object من inner هي :

- 1- `outer o=new outer();`
`outer.inner i1=o.newinner();// newinner() is the method in line 1`
- 2- `outer o=new outer();`
`outer.inner i2=o.new inner();`
- 3- `outer.inner oo = new outer.inner()`
- 4- الإجابتان 1 و 2 .
- 5- الأجوبة السابقة جميعها صحيحة.

51 – في الكود السابق أيضاً بفرض الصف inner له private access فإنه :

- 1 - لا يمكن الاستفادة من هذا الكلاس خارج الـ `outer` .
- 2 - لا يمكن الوصول لهذا الكلاس إلا بتعريف `object` من الـ `outer` ثم تعريف `object` منه.
- 3 - يمكن الاستفادة من هذا الكلاس بتعريف تابع ضمن الـ `outer` يرد `object` من الـ `inner`.

52 – ما هو خرج البرنامج التالي :

```
class MyClass
{
    private int j=12;
    {
        j++; //1
        System.out.println(j); //2
    }
}
```

```

public class Main {

    public static void main(String[] args) {

        MyClass o=new MyClass();

    }

}

```

1 - الخرج هو 12.

2 - الخرج هو 13.

3 - يوجد Compiler Error في السطر 1.

4 - يوجد Compiler Error في السطر 2.

53 - نتيجة تنفيذ البرنامج التالي هي :

```

public class Main {

    public static void main(String[] args) {

        boolean [] first=new boolean[3];

        boolean [] second=new boolean[3];

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

            System.out.print((first[i].equals(second[i])));

    }

}

```

1- true true true

2- false false false

3- true false true

4 - نتيجة عشوائية متغيرة عند كل تنفيذ.

5 - يوجد Compiler Error و يمكن حله بجعل نوع المصفوفات Boolean بدلاً من boolean .

6 - الأجوبة السابقة جميعها خاطئة لأنه لا يمكن استعمال equal مع boolean .

54 - ما هو خرج البرنامج التالي :

```
public class Main {  
    void test()throws IOException  
    {  
        System.out.println("test 1");  
    }  
    public static void main(String[] args) {  
        Main m=new Main();  
        m.test(); //1  
    }  
}
```

- 1 - البرنامج سيطبع عبارة test 1.
- 2 - سيحصل Exception أثناء التنفيذ.
- 3 - يوجد Compiler error و يمكن حله بإضافة try-catch للسطر 1.
- 4 - يوجد Compiler Error و يمكن حله بإضافة throws IOException للتابع main .
- 5 - الإجابتان 3 و 4 .
- 6 - بالإضافة للإجابتين 3 و 4 يمكن حل الخطأ بإضافة throws IOException للصف Main .

55 - هل يوجد أخطاء في البرنامج التالي :

```
class father  
{  
    public father() {  
        System.out.println("father constructor");  
    }  
}
```

```

    }
}
abstract class abs extends father{ //1
    public abs() { //2
        System.out.println("abs constructor");
    }
}
class son extends abs
{
    public son()
    {
        System.out.println("son constructor");
    }
}
public class Main {
    public static void main(String[] args) {
        son s=new son();
    }
}

```

1 – يوجد خطأ لأن الصف المجرد لا يمكن أن يحوي بانياً (السطر 2) .

2 – يوجد خطأ لأن الصف المجرد لا يمكن أن يرث من صف غير مجرد (السطر 1) .

3 – لا يوجد خطأ و الخرج هو :

father constructor

abs constructor

son constructor

4 – لا يوجد خطأ و الخرج هو (حيث لن يتم المرور على الباني في الصف abs لأنه مجرد) :

father constructor

son constructor

5 – الإجابتان 1 و 2 .

56 – هل يوجد أخطاء في البرنامج التالي : ؟

```
interface my_interface{
    void print(int x)throws IOException;
}
class my_class implements my_interface
{
    public void print(int x)
    {
        System.out.println("x= "+x);
    }
}
public class Main {
    public static void main(String[] args) {
        my_class mm=new my_class();
        mm.print(3); //1
    }
}
```

1 - البرنامج سيطبع عبارة x= 3 .

2 - سيحصل Exception أثناء التنفيذ.

3 - يوجد Compiler error و يمكن حله بإضافة try-catch للسطر 1.

4 - يوجد Compiler Error و يمكن حله بإضافة throws IOException للتابع main .

5 - الإجابتان 3 و 4 .

6 - بالإضافة للإجابتين 3 و 4 يمكن حل الخطأ بإضافة throws IOException للتابع print في الصف my_class .

57 – ما هي نتيجة تنفيذ البرنامج التالي :

```
class test
{
    void print()
    {
        System.out.println("test..print");
    }
}

class test_son extends test
{
    void print () throws IOException{
        System.out.println("test2..print");
        throw new IOException();
    }
}

public class Main {
    public static void main(String[] args) {
        test_son t=new test_son();
        t.print(); //1
    }
}
```

1 – يوجد Compiler Error لأنه لا يمكن للصف son أن يقوم بعملية override للتابع print مع إضافة throws IOException لهذا التابع.

2 – يوجد Compiler Error في السطر 1 .

3 – كان من الممكن أن يكون البرنامج سليماً من الأخطاء و قابلاً للتنفيذ لو أن الـ Exception الذي في التابع print كان من النوع RuntimeException .

4 – البرنامج سليم من الأخطاء و لكن يحصل فيه IOException أثناء التنفيذ.

5 – الإجابتان 1 و 3 .

6 – الإجابات 1 و 2 و 3 .

58 – نتيجة تنفيذ البرنامج التالي هي :

```
class GeometricObject {  
    public GeometricObject() { System.out.println("A");}  
    public GeometricObject(float f, int i) { System.out.println("B");}  
}  
  
class Circle9 extends GeometricObject{  
    public Circle9() {  
        this(1);  
        System.out.println("C");  
    }  
    public Circle9(int i) {  
        this(1,2,3);  
        System.out.println("D");  
    }  
    public Circle9(int i,float f,double d) {  
        super(f,i);  
        System.out.println("E");  
    }  
}
```

```

public class Main {

    public static void main(String[] args) {

        Circle9 c = new Circle9();

    }

}

```

ABCD - 1

BEDC - 2

BACD - 3

AEDC - 4

59 – نتيجة تنفيذ البرنامج التالي هي :

```

class Test1 {

    public static void main (String[] args) { xmethod(5);}

    public static void xmethod(int length) {

        if(length > 1){

            System.out.println(length-1 + " " );

            xmethod(length - 1);

        }

    }

}

```

```

public class Main {

    public static void main (String[] args) { xmethod(5);}

    public static void xmethod(int length) {

        while(length > 1){

            System.out.println(length-1 + " " );

            xmethod(length - 1);

        }

    }

}

```

```
}  
}  
}
```

1- 4 3 2 1.

2- 4 3 2 1 1 1 1 1.

3- 5 4 3 2 1.

4- 1 2 3 4 5.

5- 4 3 2 1 (ثم يتم طباعة العدد 1 دون توقف)

60 – نتيجة تنفيذ البرنامج التالي هي :

```
public class Main {  
    public static void main (String[] args) {  
        System.out.println("Before try");  
        try{}  
        catch(java.io.IOException e ) {}  
        System.out.println("done");  
    }  
}
```

1 - يوجد Compiler Error لأنه لم يتم رمي IOException داخل تعليمة try .

2 - يوجد Compiler Error لأنه لا يوجد تعليمة finally .

3 - البرنامج صحيح و يطبع عبارة Before try .

4 - البرنامج صحيح و يطبع عبارة Before try ثم عبارة done .

61 – نتيجة تنفيذ البرنامج التالي هي :

```
public class Main {  
    public static void main (String[] args) {  
        int [] array = {1,2,3,4,5};
```

```

reverse(array);

for(int i = 0;i<array.length;i++) {
    System.out.println(array[i]+ " ");
}
}

static void reverse (int []a) {
    int aa[] = a;
    for(int i = 0;i<a.length;i++){
        aa[i] = a[a.length-1-i];
    }
}
}
}

```

1 - يوجد Compiler Error و البرنامج غير قابل للتنفيذ .

2 - البرنامج يطبع 5 4 3 2 1

3 - البرنامج يطبع 1 2 3 2 1

4 - البرنامج يطبع 5 4 3 4 5

5 - البرنامج يطبع 5 4 3 4 5 ثم يتوقف بسبب `ArrayIndexOutOfBoundsException`

62 – إن نتيجة تنفيذ البرنامج التالي هي :

```

public class Main {
    int i = getJ(); //1

    int getJ(){
        return j;
    }

    int j=10;
}

```

```
public static void main(String[] args) {  
    System.out.println(new Main().i); //2  
}  
}
```

1 - يوجد Compiler Error بسبب عملية forward reference غير مسموحة في السطر 1.

2 - يوجد Compiler Error في السطر 2.

3 - البرنامج صحيح و يطبع 0.

4 - البرنامج صحيح و يطبع 10.

5 - الإجابتان 1 و 2 .

الأجوبة :

3 - 1

2 - 2

1 - 3

4 - 4

4 - 5

3 - 6

2 - 7

4 - 8

3 - 9

4 - 10

3 - 11

5 - 12

5 - 13

5 - 14

1 - 15

1 - 16

3- 17

5- 18

5- 19

3- 20

1- 21

1- 22

4- 23

2- 24

2- 25

1- 26

4- 27

4- 28

2- 29

1- 30

4- 31

4- 32

3- 33

1- 34

1- 35

5- 36

3- 37

3- 38

2- 39

6- 40

1- 41

5- 42

4-43

2-44

1-45

2-46

4-47

2-48

4-49

4-50

3-51

2-52

5-53

5-54

3-55

5-56

6-57

2-58

5-59

1-60

4-61

3-62

ملاحظات :

- الهدف من السؤال 8 هو معرفة أنه تم ارتكاب ثلاثة أخطاء في الكود و هي :

1 - الباني لا يمكن أن يكون static

2 - لا يمكن استخدام this في تابع static

3 - يجب أن تكون this أول تعليمة في الباني عندما تشير لباني آخر.

و لكن الكومبايلر لا يظهر لنا الخطأ الثاني لأسباب تتعلق ببناء الكومبايلر لا علاقة لنا بها الآن.

- في السؤال 43 لا يمكن توقع الخرج لأن كلاً من out و err لديه Buffer خاص به و يتم إفراغ هذا الـ Buffer بشكل غير منتظم (أي في كل مرة يتم إفراغ أحدهما قبل الآخر) و هذا يؤدي إلى ظهور ترتيب مختلف للجمل في كل مرة.

لأي ملاحظات أو تنبيه عن وجود أخطاء يمكن التواصل على

m-khaled89@hotmail.com

HammooD

Damascus university 2010