Faculty of Electronic Technology | Name

Sem 4

Course: CS 213 OOP1 | Student No.

Midterm Exam Summer 2019 | Date

ممنوع استعمال الهاتف النقال خلال الامتحان

Q1 اختر الاجابة الصحيحة: **Choose the right answer (5 درجات)**

1. Which of the following declarations could be correct in Java (where ... represents some code)? أي من التعريفات الاتية يعتبر صحيح في لغه الجافا (حيث ان النقاط تمثل كود برمجي)

a) class C extends A, B { ... }
b) interface C implements A, B { ... }
c) interface C extends A, B { ... }
d) class A {...} class B { ...} class C implements A, B { ... }

2. Calling a method through Object is known as استدعاء داله عن طريق الكائن تعرف ب

a) Message رساله (b) Method header عنوان الداله (c) Method signature توقيع الدالة
(d) All of the above كل الاجابات السابقه

3. Consider the following statement which is defined in a class (let's call it class Test). The keyword static:
private static int serial = 0;

a) Means that serial is a constant.
b) Ensures that only one instance of serial exists and it will be shared by all objects of type Test.
c) Means that serial should be capitalized (e.g. SERIAL) to comply with Java naming conventions.
d) Results in a syntax error because it is missing the keyword final.

4. بالنظر إلى التعريف التالي للطيور والدجاج ، أي من العبارات المذكورة لن تنفذ؟

4. Given the following definition of Bird and Chicken, which of the given statements will not compile?

```
abstract class Bird implements Livestock {}
class Chicken extends Bird {}
```

a) Bird bird = new Chicken();
b) Livestock livestock = new Chicken();
c) Bird bird = new Bird();
d) None of these will compile

5. ما هو نوع العلاقة الموجودة بين الدالة someMeth في الفئة Aو الدالة some Meth في الفئة B

5. What type of relationship exists between someMeth in classes A and someMeth in class B?

```
class A {
  private void someMeth() {
      System.out.println( "from class A" ); } }
class B extends A {
  public void someMeth( String x ) {
  System.out.println( "from class B: " + x ); } }
```

(a) method overriding (b) method overloading (c) both method overriding and method overloading (d) neither method overriding nor method overloading

Q2 من الكود التالي giving the following code (5 درجه)

```
class SuperClass {
   private int x;
   private int y;
   public SuperClass ( )
      { x = 2;  y = 3; }
   public SuperClass (int x, int y)
      { this.x = x;  this.y = y; }
   public String toString ( )
      { return "Numbers are: " + x + " and " + y; }
   public  int  returnSum( )
      { return (x+y); }
}
class SubClass extends SuperClass {
   private int z;
   public SubClass( )
      { super( ); }
   public SubClass(int x, int y)
      { super(x, y);  z = 4; }
   public int returnSum( )
      { return (super.returnSum() + z); }
}
```

and the declarations:

SuperClass obj1 = new SuperClass();
SuperClass obj2 = new SubClass(1, 2);

What will the following statements display to the screen? الناتج من الجمل الاتية بناءا علي الكود البرمجي السابق

a) System.out.println (obj1); *Numbers are: 2 and 3*
b) System.out.println(obj2); *Numbers are: 1 and 2*
c) System.out.print("Sum is:" + obj2.returnSum()); *Sum is: 7*

Q3 وضح الفرق بين كل من بمثال: (5 درجات)

1. Interface الواجهة و التجريد Abstraction

Abstract class vs Interface (Different)

Abstract class	Interface

• To declare an abstract class, use `abstract` keyword.
`public abstract class B{`
`}`
	• To declare an interface, use `abstract` keyword.
	`public interface B{`
	`}`

• A class can extend only one abstract class.
`class A extends B{`
`}`
	• A class can implement more than one interface.
	`class A implements C, D, E{`
	`}`

• In relationship, we say A is B.	• In relationship, we A has C, D, and E.

2. static polymorphism تعدد الوجوه dynamic polymorphism

Difference between Static & Dynamic Polymorphism

Static Polymorphism	Dynamic Polymorphism
It relates to method overloading.	It relates to method overriding.
Errors, if any, are resolved at compile time. Since the code is not executed during compilation, hence the name static.	In case a reference variable is calling an overridden method, the method to be invoked is determined by the object, your reference variable is pointing to. This is can be only determined at runtime when code in under execution, hence the name dynamic.
Ex:	Ex:

```
void sum (int a , int b);
void sum (float a, double b);
int sum (int a, int b); //compiler gives erro
r.
```

```
//reference of parent pointing to child objec
t
 Doctor obj = new Surgeon();
// method of child called
obj.treatPatient();
```

3. One dimensional Array and String السلسله الحرفية والمصفوفه ذات البعد الواحد

تركيبة من تراكيب البيانات لتمثيل مجموعه من البيانات ذات النوع الواحد، بمعني اخر هي مجموعه من خلايا الذاكرة المتجاورة ذات الاسم الواحد وتحمل بيانات ذات نوع واحد

ARRAY VERSUS STRING

Array	String
Arrays are a sequential collection of elements of similar data types.	Strings refer to a sequence of characters represented as a single data type.
Elements of arrays are stored contiguously in increasing memory locations.	Strings can be stored in any manner in memory locations.
An array is a special variable that can hold more than one value at a time.	Strings can only hold char data which are the most commonly used data types.
Arrays are mutable meaning the fields can be modified.	Strings are immutable meaning the value cannot be changed in memory once created.
The length of an array is fixed.	The size of a string is not fixed.

DifferenceBetween.net

Summary of Array vs. String

Strings and arrays are quite similar except the length of an array is fixed whereas strings can have a variable number of elements. Technically, arrays are a special type of variable that can hold more than one value at a time. They are a sequential collection of elements of similar data types, whereas strings are a sequence of characters used to represent text rather than numbers. In addition, strings are immutable which means the value of an object cannot be modified once it's created, while arrays are mutable meaning the fields can be modified. Simply put, an array is a collection of like-type variables whereas a string is a sequence of characters represented by a single data type.

4. Continue and break in If statements

5. تستخدم لتغيير مسار البرنامج والانتقال من داخل الحلقة الى خارج الحلقة.

مهمتها الاستمرار في توجيه التحكم إلى نهاية الحلقة التكرارية والرجوع إلى بدايتها واكمال تنفيذ الحلقة حتى نهايتها

6. Local Variables & Instance Variables المتغيرات اللحظية والمتغيرات المحلية

A variable defined within a block or method or constructor is called local variable. These variable are created when the block in entered or the function is called and destroyed after exiting from the block or when the call returns from the function.

Declared in a class outside any method, constructor or block. Instance Variable can be accessed only by creating objects.

Q4 (4 درجة) من الكود البرمجي التالي اوجد 3 أخطاء لغوية Syntax errors و \ أو اخطاء منطقية logical errors مع التصحيح:

Q4. Identify the errors in the following codes and recommend the corrections: Circle each error and suggest a correction.

```
public class Hobbit extends Actor {
 public static constant double MAX_STEALTH = 100.0;
 private double stealth;
      no return type on constructor        final not constant

 public void Hobbit() {
  stealth = MAX_STEALTH / 2.0;
  }

 public void setStealth(double stealth) {
  stealth = stealth;
  }
                          Calls itself, need super displayStatus()
 public void displayStat..() {
  displayStatus();
  System.out.printf(" Stealth: %d", stealth);
  }
     Missing String return type.
 public toString() {
  return String.format("%s Stealth:%4.1f",
       super.toString(), stealth);
 }
} // end class Hobbit
```

Q5ما هو المخرج من الكود التالي (4 درجات)

```java
public class Stuff {
   private static final int n = 2;
   private String string;
   private int num;

   public Stuff(String s, int num) {
       this.num = num;
       string = s;
   }

   public void doSomething(double d) {
       d = d * num;
       System.out.println(this);
   }

   public double changeSomething(String s) {
       s = string;
       return n * num;
   }

   public String toString() {
       return string + " has " + num;
   }
}
```

```java
public class TestStuff {
   public static void main(String[] args) {
       Stuff s = new Stuff("in", 5);
       System.out.println(s);
       double doubleValue = 2.5;
       s.doSomething(doubleValue);
       System.out.println(doubleValue);
       s = new Stuff("more", 3);
       String str = "word";
       System.out.println(s.changeSomething(str));
       System.out.println(s);
       System.out.println(str);
   }
}
```

```
Program Output
in has 5
2.5
6.0
more has 3
word
```

Q6 (7 درجات)

Design UML class diagram and implement java code that can represent Animals system. Animals have two behaviors; they can **speak()** and they can **move()**. By default, when an animal moves, the text "**This animal moves forward**" is displayed. By default, when an animal speaks, the text "**This animal speaks**" is displayed. A general Animal should not be able to be instantiated. Define also two classes, **Goose** and **Lynx** that are Animals. Both Goose and Lynx behave such that where "animal" is displayed in **speak()** or **move()**, "**goose**" or "**lynx**" is displayed by the appropriate classes. Finally, any instance of Goose can **fly()**, just as any Flying object can. An Airplane is also a **Flying object**. Define the **Airplane class** such that it is Flying and make sure that any instance of Goose is also **flying**. Instances of either Goose or Airplane should be using same fly() method from **flying**.

ارسم مخطط الفئة ثم حوله لكود جافا برمجي لبرنامج نظام Animals. الحيوانات لها سلوكان. يمكنهم التحدث **speak()** ويمكنهم التحرك **move()**. افتراضيًا ، عندما يتحرك حيوان ، يتم عرض النص "يتحرك هذا الحيوان للأمام" وعندما يتحدث حيوان ، يتم عرض النص "يتحدث هذا الحيوان". يجب ألا يكون الحيوان Animal العام قادرًا على إنشاء مثيل له. حدد أيضًا فئتين ، هما أوزة **Goose** ولينكس **Lynx** وهما من الحيوانات. يتصرف كلاً من أوزة ولينكس بحيث يتم عرض "الحيوان" في الكلام **speak()** أو التحرك **move()** على انه أوزة "**Goose**" أو الوشق "**lynx**" في الفئات المناسبة. أخيرًا ، يمكن لأي مثيل من Goose الطيران **fly()** ، تمامًا كما يمكن لأي كائن **Flying**. الطائرة هي أيضا كائن تحلق **flying**. حدد فئة **Airplane** بحيث تكون Flying وتأكد من أن أي مثيل من Goose هو **Flying** أيضًا. مثيلات Goose أو Airplane هيا ايضا **Flying**. يجب أن تستخدم مثيلات إما الأوز أو الطائرة نفس دالة يطير **fly()** من طيران **Flying**.

```
abstract class Animal
{
private final String name;
public Animal( String name ){
  this.name = name;}
public Animal()
{ this( "animal" );}
public void speak()
{ System.out.println( "This " + name + " speaks" );}
public void move()
{ System.out.println( "This " + name + " moves forward" ); }
}
class Lynx extends Animal
{
  public Lynx(){
  super( "lynx" );  }
}
interface Flying {
  public void fly();
}
class Goose extends Animal implements Flying
{
  public Goose(){
    super( "goose" ); }
  public void fly() {
   System.out.println( "This " + getClass().getName() +" soars, wings
flapping.");
}
class Airplane implements Flying {
public void fly() {
System.out.println( "This " + getClass().getName() +
" soars, engines running.");
}
}
}
```

تمنياتي للجميع بالتوفيق ☺