Exercicis LTP

Tema 1: Introducció

Part I: Qüestions

1. Els següents programes Java contenen errors. Assenyala les sentències que contenen errors. Explica en què consisteix cada error i com podria solucionar-se.

Programa 1:

```
public class Exercise1 {
  public static void main(String[] args) {
    A objA = new A();
    System.out.println("in main(): ");
    System.out.println("objA.a ="+objA.a);
    objA.a = 222;
  }
}
```

```
public class A {
    private int a = 100;
    public void setA(int value) {
       a = value;
    }
    public int getA() {
       return a;
    }
}//class A
```

Programa 2:

```
public class Exercise2 {
  public static void main(String[] args) {
    System.out.println("in main(): ");
    System.out.println("objA.a ="+getA() );
    setA(123);
  }
}
```

```
public class A {
    private int a = 100;
    public void setA(int value) {
       a = value;
    }
    public int getA() {
       return a;
    }
}//class A
```

2. Donat el següent programa, assenyala el (o els) error(és) que puga haver-hi en les últimes quatre assignacions de la classe `PolymorphicAssignment`.

```
public class ClassA {
}// end ClassA

public class ClassB extends ClassA {
}// end ClassB
```

```
public class PolymorphicAssignment {
  public static void main(String[] args) {
    ClassA obj1 = new ClassA();
    ClassA obj2 = new ClassA();
    ClassB obj3 = new ClassB();

    obj1 = obj2;
    obj1 = obj3;
    obj3 = obj2;
    obj3 = obj1;
  } //end main
} //end class
```

3. Vertader o fals?:

   (a) Una subclasse és generalment més xicoteta que la seua superclasse.
   (b) Un objecte d'una subclasse és també un objecte de la seua superclasse.
   (c) Tots els mètodes en una classe abstracta han de declarara abstractes.
   (d) En un programa Java no pot haver-hi dos mètodes amb el mateix nom.
   (e) Una subclasse d'una classe abstracta que no implementa tots els mètodes abstractes que hereta ha de ser abstracta.
   (f) Tots els mètodes en una super classe abstracta han de ser abstractes.

4. Donat el segent programa Java, indica quins tipus de polimorfisme s'usen en cada cas. Els casos estan identificats per nombres. Quan dos fragments de codi tenen el mateix nombre és perquè és necessari mirar les dues parts per a determinar el tipus de polimorfisme.

```
public class Polymorphism {

//*************************************
// 5.
//*************************************
  abstract class Car {
    public void getBrand(){};
  }

  class Polo extends Car {
    String brand = "Volkswagen";
    public void getBrand(){
      System.out.println();
      System.out.println(brand);
    };
  }

  public void carBrand(Car c){
    c.getBrand();
  }

  public static void main(String[] args){
    Polymorphism o = new Polymorphism();

//*************************************
// 1.
//*************************************
    System.out.println("Hellp "+"world");
    System.out.println(1+2);
    System.out.println();

//*************************************
// 2.
//*************************************
    int x1=1,y1=1;
    o.add(x1,y1);
    float x2=1,y2=1;
    o.add(x2,y2);

//*************************************
// 3.
//*************************************
    int x3=2,y3=1;
    o.subt(x3,y3);

//*************************************
// 4.
//*************************************
    // Create arrays of Integer
    // Double and Character
    Integer[] intArray = {1,2,3,4,5};
    Double[] doubleArray = {1.1,2.2,3.3};
    Character[] charArray = {'H','E','Y'};

    System.out.println("intArray contains:");
    o.printArray(intArray);

    System.out.println("\ndoubleArray contains:");
    o.printArray(doubleArray);

    System.out.println("\ncharArray contains:");
    o.printArray(charArray);

//*************************************
// 5.
//*************************************
    Polo m = o.new Polo();
    o.carBrand(m);
  } // END MAIN

//*************************************
// 2.
//*************************************
  public void add (int x, int y) {
    System.out.println("Integer addition: ");
    System.out.println(x+y);
    System.out.println();
  }

  public void suma (float x, float y) {
    System.out.println("Double addition: ");
    System.out.println(x+y);
    System.out.println();
  }

//*************************************
// 3.
//*************************************
  public void resta (float x, float y) {
    System.out.println("Double substraction: ");
    System.out.println(x-y);
    System.out.println();
  }

//*************************************
// 4.
//*************************************
  public <E> void printArray(E[] inputArray) {
    for (E element : inputArray){
        System.out.printf("%s ", element);
    }
    System.out.println();
  }
} // END CLASS Polymorphism
```

Output:

```
Hello world
3

Integer addition:
2

Double addition:
2.0

Double substraction:
1.0

integerArray contains:
1 2 3 4 5

doubleArray contains:
1.1 2.2 3.3

characterArray contains:
H E Y

Volkswagen
```

5. Quin tipus de polimorfisme està present en el següent fragment de codi?

```
public int  add(int x, int y){
    return x + y;
}
public String  add(String s, String t){
    return s + t;
}
```

6. Vertader o fals?:

   (a) Tots els llenguatges de programació permeten la reflexió.

   (b) La reflexió pot usar-se per a observar i modificar el programa en temps d'execució.

   (c) Els llenguatges orientats a objectes no suporten la reflexió.

   (d) La reflexió és una estratègia clau per al polimorfisme.

7. Després d'executar aquest codi, la variable "x" val 2 i la variable "y" val 2. Quin mecanisme de pas de paràmetres s'ha utilitzat?

```
int funcion(int a, int b)
{
    a++;
    return b;
}

int x = 1, y = 2;
y = funcion(x, 2 * x);
```

8. Donat el següent programa:

```
static void foo(int a, int b) {
  a = a+b;
}
public static void main () {
  int x = 0
  int y = 10
  foo(x,y)
}
```

   indica quin és el valor de les variables `x` i `y` en acabar l'execució de `main` seguint:

   (a) l'estratègia de pas de paràmetres per referència.

   (b) l'estratègia de pas de paràmetres per valor.

9. Donat el següent programa:

```
public class Exercise9 {
    public static void main(String[] args) {
       CallByAny cbn = new CallByAny();
       int x = 2, y;
       y = cbn.foo(x, x+1);
       System.out.println("x: " + x);
       System.out.println("y: " + y);
    }
}
class CallByAny {
     int foo(int a, int b){
         a++;
         return (a+b);
     }
}//class CallByAny
```

(a) Tenint en compte que Java utilitza un mecanisme de pas de paràmetres per valor (*call by value*), indica el resultat de l'execució d'aquest programa.

(b) Quin seria el resultat si el mecanisme de pas de paràmetres fora per referència (*call by reference*)?

(c) Quin seria el resultat si el mecanisme de pas de paràmetres fora per necessitat (*call by need*)?

10. Donat el següent programa:

```
public class Exercise10 {
    public static void main(String[] args) {
       CallByAny cba = new CallByAny();
       int x = 3, y;
       y = cba.foo(x, x+2);
       System.out.println("x=" + x);
       System.out.println("y=" + y);
    }
}
class CallByAny {
     float foo(int a, int b){
         a +=10;
         return b;
     }
}//class CallByAny
```

(a) Tenint en compte que Java utilitza un mecanisme de pas de paràmetres per valor (*call by value*), indica el resultat de l'execució d'aquest programa.

(b) Quin seria el resultat si el mecanisme de pas de paràmetres fora per referència (*call by reference*)?

(c) Quin seria el resultat si el mecanisme de pas de paràmetres fora per necessitat (*call by need*)?

11. Donat el següent programa:

```
public class Classe1 {
    public static void main(String[] args){
        Example ex = new Example();
        int x = 3, y;
        y = ex.foo(x, x+2);
        System.out.println("x=" + x);
        System.out.println("y=" + y);
        }
}
class Example{
        int foo(int a, int b){
             a+=b;
        return a;
        }
}
```

(a) Tenint en compte que Java utilitza un mecanisme de pas de paràmetres per valor (*call by value*), indica el resultat de l'execució d'aquest programa.

(b) Quin seria el resultat si el mecanisme de pas de paràmetres fora per referència (*call by reference*)?

12. Donat el següent programa:

```
public class Classe {

    public static void main(String[] args){
       CallByName cbn = new CallByName();
       float x = 3, y;
       y = cbn.metodo(x, 1/x);
       System.out.print("y=" + y);
       System.out.println("x=" + x);
}
}
```

```
class CallByName{
      float metodo(float contador, float incremento){
         float sum = 0;
         for (contador=1;contador<=3;contador++){
             sum = sum + incremento;
              }
          return sum;
  }
}
```

(a) Tenint en compte que Java utilitza un mecanisme de pas de paràmetres per valor (*call by value*), indica el resultat de l'execució d'aquest programa.

(b) Quin seria el resultat si el mecanisme de pas de paràmetres fora per referència (*call by reference*)?

(c) Quin seria el resultat si el mecanisme de pas de paràmetres fora per necessitat (*call by need*)?

13. Vertader o fals?

(a) En l'abast estàtic l'àmbit d'una variable és el fragment de codi sintàcticament més pròxim a la seua declaració.

(b) El recollector de basura (*garbage collector*) s'encarrega de l'assignació i alliberament automàtic dels recursos de memòria per a un programa.

14. Donat el següent programa:

```
1    program scoping
2    var x: integer;

3    procedure one;
4     var x: integer;
5     begin
6        x:=13;
7        three;
8        end;          // end one

9    procedure two;
10    var x: integer;
11    begin
12       x:= 12;
13       three;
14       end;          // end two
```

```
15     procedure three;
16       begin
17         writeln(x);
18       end;          // end three

19     begin           // begin main
20       x:= 14;
21       one;
22       two;
23     end
```

(a) Quin és el resultat de l'execució del programa `scoping` considerant abast estàtic?

(b) Quin és el resultat de l'execució del programa `scoping` considerant abast dinàmic?

15. Donat el següent programa:

```
1.   n: integer            --- global declaration
2.   procedure first
3.       n:=1
4.   procedure second
5.       n: integer     --- local declaration
6.       first()
7.   n:=2
8.   if read_integer() > 0
9.       second()
10. else
11.      first()
12. write_integer(n)
```

(a) Quin és el resultat d'executar aquest programa considerant abast estàtic?

(b) Quin és el resultat d'executar aquest programa considerant abast dinàmic?

16. Vertader o fals?

(a) Si un llenguatge permet recursió, les variables locals dels subprogrames recursius no poden emmagatzemar-se estàticament però així i tot poden emmagatzemar-se en una pila.

(b) El recollector de basura (*garbage collector*) s'encarrega de l'alliberament automàtic dels recursos de memòria per a un programa.

(c) Una pila és un tipus d'emmagatzematge que segueix un ordre "primer a entrar, primer a eixir".

(d) Un heap o monticle és una regió d'emmagatzematge en la qual els subblocs de memòria són assignats i alliberats en temps fixos.

17. Indica el paradigma al que pertanyen les següents característiques:

(a) L'assignació destructiva.

(b) Les variables lògiques.

(c) Ordre superior.

(d) Les instruccions de control (while, if, for,...).

Parte II: Test

18. El següent programa Java:

```
int y;
x = 42+y;
```

- [A] És correcte ja que Java té tipificació implícita.
- [B] És incorrecte: la variable `y` no ha sigut declarada.
- [C] És incorrecte: la variable `x` no ha sigut declarada.
- [D] És correcte gràcies a la inferència de tipus de Java.

19. Els tipus es descriuen mitjançant un llenguatge d'expressions de tipus. Indica quin de les següents afirmacions és **FALSA**:

- [A] Els tipus bàsics o primitius no formen part de les expressions de tipus.
- [B] Les variables de tipus representen tipus.
- [C] Els constructors de tipus s'usen per a obtenir nous tipus.
- [D] Les regles de construcció d'expressions de tipus descriuen com construir noves expressions de tipus.

20. ¿Quin és la utilitat dels tipus en un llenguatge de programació?

- [A] La seua principal utilitat és que, si el programa no té errors de tipus, es garanteix que no hauran errors d'execució..
- [B] Els tipus són essencials per a poder definir una semàntica dinàmica del llenguatge de programació.
- [C] Els tipus ajuden a detectar errors de programació.
- [D] Els tipus no serveixen per res.

21. El següent programa Java:

```
public class C {
  int exemple(int x, int y) {...}
  void exemple(char x) {...}
}
```

- [A] És un exemple de declaració de mètodes amb polimorfisme universal.
- [B] És un exemple de declaració de mètodes sobrecarregats (polimorfisme ad-hoc).
- [C] És incorrecte: la variable `x` s'utilitza amb tipus diferents en dues funcions diferents.
- [D] És un exemple de declaració de mètodes genèrics.

22. Quin de les següents declaracions defineix una classe genèrica en Java?

- [A] `public class foo<T> { ... }`.
- [B] `public class foo<T k> { ... }`.
- [C] `public <T> class foo(){ ... }`.
- [D] `public <T> class foo { ... }`.

23. Pel que fa a la següent definició de mètode genèric, indica l'opció **CORRECTA**:

```
public static < E > void printArray( E[]  inputArray )
{    ...   }
```

A La variable genèrica `E` ha d'haver sigut declarada en la classe que conté a aquest mètode genèric. Per exemple així:
```
class Generica {
   Figura E;
   public static < E > void printArray( E[] inputArray ){...}
   ...
}
```

B La variable genèrica `E` ha d'haver sigut declarada en la trucada a aquest mètode genèric. Per exemple així: `printArray(Figura E)`.

C Aquest mètode genèric ha de pertànyer a una classe genèrica o a una classe que herete d'una classe genèrica.

D Encara que siga genèric, aquest mètode pot pertànyer a una classe no genèrica.

24. Donades les següents definicions de classes

```
public class Animal {
}
public class Dog extends Animal {
}
```

Indica quin de les següents assignacions és **INCORRECTA**:

A
```
Animal animal1 = new Dog();
```

B
```
Animal animal1 = new Dog();
Dog animal2 = (Dog) animal1;
```

C
```
Dog animal1 = new Animal();
```

D
```
Animal animal1 = new Dog();
Animal animal2 = animal1;
```

25. Indica quin de les següents afirmacions sobre la reflexió és **FALSA**:

A Es tracta d'una característica que permet inspeccionar i/o modificar l'execució d'un programa en temps d'execució.

B Tots els llenguatges de programació tenen alguna biblioteca o conjunt de funcions que proporcionen reflexió.

C Resulta útil en tasques de metaprogramació.

D Si s'usa malament, pot afectar al rendiment i seguretat del programa objecte.

26. En un llenguatge de programació imperatiu:

A No hi ha efectes laterals.

B No hi ha tipus.

C El programa és una seqüència d'instruccions que pot canviar el seu estat.

D El programa consisteix en una descripció lògica del problema a resoldre.

27. Indica quina de les següents afirmacions és **CIERTA**:

A La lògica d'un programa declaratiu s'expressa usant instruccions de control.

B Un programa declaratiu pot entendre's com una especificació executable d'un problema.

C La programació en el paradigma imperatiu es basa en la definició d'equacions.

D En el paradigma lògic i en el paradigma funcional es donen efectes laterals tal com ocorre en el paradigma imperatiu.

28. Indica quin de les següents afirmacions sobre paradigmes és **FALSA**:

A. En els paradigmes basats en interacció, un programa és una comunitat d'entitats que interactuen seguint certes regles d'interacció.

B. L'abstracció, l'encapsulament, la modularidad i la jerarquia són elements fonamentals del paradigma orientat a objectes.

C. En el paradigma de programació dirigida per esdeveniments, el flux del programa està determinat per esdeveniments o missatges.

D. L'objectiu de la programació parallela consisteix a controlar el flux del programa mitjançant esdeveniments.

29. Indica quin de les següents afirmacions referents a la Programació Declarativa (PD) i Imperativa (PI) és **FALSA**:

A. Les instruccions d'un programa declaratiu són un conjunt d'inferències lògiques.

B. Un programa en PD es correspon amb l'especificació d'un problema.

C. El model de computació de la PI és una màquina d'estats.

D. Les variables en la PI representen referències a la memòria.

30. Indica quina de les següents afirmacions és **CIERTA**:

A. un desavantatge dels llenguatges de programació declaratius és que, per a resoldre un mateix problema, es requereixen en general més línies de codi que usant altres llenguatges de programació més convencionals, com Pascal.

B. al ser de major nivell, els programes declaratius resulten més fàcils de mantenir que els corresponents programes imperatius.

C. no existeix cap llenguatge orientat a objectes i alhora declaratiu.

D. els llenguatges declaratius manquen d'aplicacions pràctiques.

31. Indica quina de les següents associacions entre llenguatge de programació i concepte (o propietat) inherent al mateix és **CIERTA**:

A. llenguatge imperatiu - herència

B. llenguatge funcional - polimorfisme

C. llenguatge lògic - estats explícits

D. llenguatge imperatiu - ordre superior

32. Indica quina de les següents associacions és **FALSA**:

A. Paradigma imperatiu $\Leftrightarrow$ l'execució del programa consisteix en construir els estats de la màquina necessaris per a arribar a la solució.

B. Paradigma declaratiu $\Leftrightarrow$ les instruccions són fórmules d'algun sistema lògic.

C. Paradigma imperatiu $\Leftrightarrow$ se satisfà el principi "PROGRAMA = ESPECIFICACIÓ D'UN PROBLEMA".

D. Paradigma declaratiu $\Leftrightarrow$ se satisfà el principi "PROGRAMA = LÓGICA + CONTROL".

33. Indica quina de les següents afirmacions és **CERTA**:

A. L'inici de la Programació Concurrent està en la invenció de la interrupció a la fi dels 50.

B. En la programació imperativa es redueix el gap semàntic entre especificació i programació.

C. En el paradigma declaratiu el concepte bàsic és l'estat, que es defineix pels valors de les variables involucrades en cada punt de la computació.

D. Java és un llenguatge de programació sense herència.