Exercises LTP

Theme 1: Introduction

Part I: Questions

1. The following Java applications contain errors. Point out the statement(s) that contain errors. Explain what each of the errors is, and how it can be fixed.

Program 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
```

Program 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. Given the following Java program, point out the error(s) (if any) in the last four assignments in method `main` of class `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. TRUE/FALSE?:

   (a) A subclass is generally smaller than its superclass.

   (b) A subclass object is also an object of the superclass.

   (c) All methods in an abstract superclass must be declared abstract.

   (d) A Java program can not contain two methods with the same name.

   (e) A subclass B of an abstract class A that does not implement all abstract methods of A must be declared abstract.

   (f) All methods in an abstract superclass must be declared abstract.

4. Given the following Java program, which kind of polymorphism is used in each case? Note that the different cases have a numeric identifier. When two fragments of code have the same number you need to consider both to discover which kind of polymorfism is at stake

```
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("Hello "+"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 add (float x, float y) {
    System.out.println("Double addition: ");
    System.out.println(x+y);
    System.out.println();
  }

//***************************************
// 3.
//***************************************
  public void subt (float x, float y) {
    System.out.println("Double subtraction: ");
    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:                                               Array integerArray contains:
  Hello world                                           1 2 3 4 5
  3
  Integer addition:                                     Array doubleArray contains:
  2                                                     1.1 2.2 3.3 4.4

  Double addition:                                      Array characterArray contains:
  2.0                                                   H E L L O

  Double subtraction:                                   Volkswagen
  1.0                                                 */
```

5. Which kind of polymorphism is used in the following code?

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

6. TRUE/FALSE?:

   (a) Reflection is allowed in all programming languages.
   (b) Reflection can be used to observe and modify a program during its execution.
   (c) Object-oriented programming languages do not support reflection.
   (d) Reflection is a key strategy for polymorphism.

7. After executing the following code, the value of variable "x" is 2 and the value of variable "y" is 2. Which passing parameter mechanism has been used?

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

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

8. Consider the following program:

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

   which is the final value of x and y if the parameter passing mechanism is...

(a) ...call by reference?

(b) ...call by value?

9. Consider the following program:

```
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) Since Java's parameter passing mechanism is *call by value*, which is the outcome of the program?

(b) Which is the outcome if *call by reference* is assumed instead?

10. Consider the following program:

```
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) Since Java's parameter passing mechanism is *call by value*, which is the outcome of the program?

(b) Which is the outcome if *call by reference* is assumed instead?

11. Consider the following program:

```
public class Clase1 {
    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) Since Java's parameter passing mechanism is *call by value*, which is the outcome of the program?

(b) Which is the outcome if *call by reference* is assumed instead?

12. Consider the following program:

```
public class Class {

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

```
class ParameterPassing{
      float method(float counter, float increment){
         float sum = 0;
         for (counter=1;counter<=3;counter++){
             sum = sum + increment;
              }
          return sum;
  }
}
```

   (a) Since Java's parameter passing mechanism is *call by value*, which is the outcome of the program?
   (b) Which is the outcome if *call by reference* is assumed instead?

13. True or false?

   (a) Under static scope, the scope of a variable is the fragment of code which is syntactically closer to its declaration
   (b) A programming language is "robust" if its programs written in a given computer can be executed in a different computer
   (c) The *garbage collector* deals with the automatic allocation and deallocation of memory resources for programs

14. Consider the following program:

```
program scoping
var x: integer;
procedure one;
 var x: integer;
 begin
    x:=13;
    three;
 end;//one
procedure two;
 var x: integer;
 begin
    x:= 12;
    three;
 end;//two
procedure three;
 begin
    writeln(x);
 end;//three
begin
   x:= 14;
   one;
   two;
end.
```

   (a) Which is the outcome of program `scoping` when static scope is used?
   (b) Which is the outcome of program `scoping` when dynamic scope is used?

15. Consider the following program:

```
n: integer            --- global declaration
procedure first
    n:=1
procedure second
    n: integer     --- local declaration
    first()
n:=2
if read_integer() > 0
    second()
else
    first()
write_integer(n)
```

   (a) Which is the outcome of the program when static scope is used?
   (b) Which is the outcome of the program when dynamic scope is used?

16. True or false?

    (a) In a language with recursion, local variables in recursive programs cannot be given a static allocation. Still, they can be allocated in a stack

    (b) The *garbage collector* deals with the automatic deallocation of memory during program execution

    (c) A stack follows a *first-in, first-out* storage policy

    (d) A *heap*is a storage region where memory blocks are allocated and deallocated at fixed time slots

17. Write the programming paradigm that corresponds to the following features:

    (a) Destructive assignment

    (b) Logic variables

    (c) Higher-order

    (d) Control instructions (while, if, for,...)

Part II: Test

18. The following Java program:

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

$\boxed{\text{A}}$ It is correct due to Java's implicit typing.

$\boxed{\text{B}}$ It is wrong: variable `y` is not declared.

$\boxed{\text{C}}$ It is wrong: variable `x` is not declared.

$\boxed{\text{D}}$ It is correct due to Java's type inference.

19. Types are specified by using a language for type expressions. In this setting, which of the following claims is $\boxed{\textbf{WRONG}}$?

$\boxed{\text{A}}$ The basic (or primitive) types never occur in type expressions.

$\boxed{\text{B}}$ Type variables represent types.

$\boxed{\text{C}}$ Type constructors are used to obtain new types.

$\boxed{\text{D}}$ The rules for writing type expressions describe how to build new type expressions.

20. Which is the usefulness of types in programming languages?

$\boxed{\text{A}}$ The absence of type errors guarantees the absence of execution errors.

$\boxed{\text{B}}$ Types are essential to define the dynamic semantics of programming languages.

$\boxed{\text{C}}$ Types help to detect programming errors.

$\boxed{\text{D}}$ Types are useless.

21. The following Java program:

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

$\boxed{\text{A}}$ Exemplifies the declaration of universally polymorphic methods.

$\boxed{\text{B}}$ Exemplifies the declaration of overloaded methods (ad-hoc polymorphism).

$\boxed{\text{C}}$ It is illegal: variable `x` is used with different types in different functions.

$\boxed{\text{D}}$ Exemplifies the declaration of generic methods.

22. Which of the following declarations defines a Java generic class?

$\boxed{\text{A}}$ `public class foo<T> { ... }`.

$\boxed{\text{B}}$ `public class foo<T k> { ... }`.

$\boxed{\text{C}}$ `public <T> class foo(){ ... }`.

$\boxed{\text{D}}$ `public <T> class foo { ... }`.

23. With regard to the following definition of a generic method, which is the **CORRECT** claim?

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

A The generic variable `E` must be previously declared in the class containing this generic method. For instance:

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

B The generic variable `E` must be previously declared in the call to this generic method. For instance: `printArray(Figure E)`.

C This generic method must belong to a generic class or to a subclass of a generic class.

D Although the method is generic, it can belong to a non-generic class.

24. Consider the following class definitions:

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

Which of the following assignments is **WRONG**?

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. Which of the following claims about reflection is **WRONG**?

A It is a feature enabling the observation or modification of the program during the execution.

B Every programming language provides a library or set of functions to give support to reflection.

C It is useful in metaprogramming.

D A bad use may hinder the performance or compromise security of the object program.

26. In an imperative programming language:

A Side effects are not possible.

B Using types is not possible.

C A program is a sequence of instructions that may change the program state.

D A program consists of a logic description of the problem at stake.

27. Which of the following claims is **TRUE**?

A The logic of a declarative program is expressed by means of control instructions.

B A declarative program can be seen as an executable specification of a problem.

C An imperative program consists of a set of equations.

D As for the imperative paradigm, in the logic and functional paradigms side effects also happen.

28. Which of the following claims about programming paradigms is $\boxed{\textbf{WRONG}}$?

   $\boxed{\text{A}}$ In interaction-based paradigms, programs are collections of entities that interact according to some interaction rules.

   $\boxed{\text{B}}$ Abstraction, encapsulation, modularity and hierarchy are essential ingredients of the object-oriented paradigm.

   $\boxed{\text{C}}$ In the event-driven programmign paradigm, the program flow is determined by events or messages.

   $\boxed{\text{D}}$ The goal of parallel programming is controlling the program flow by means of events.

29. Which of the following claims about Declarative Programming (DP) and Imperative Programming (IP) is $\boxed{\textbf{WRONG}}$?

   $\boxed{\text{A}}$ The instructions of a declarative program are a set of logic inferences.

   $\boxed{\text{B}}$ A declarative program corresponds to the specification of a problem.

   $\boxed{\text{C}}$ The computational model of imperative programming is a state machine.

   $\boxed{\text{D}}$ Variables in imperative programs represent memory references.

30. Which of the following claims is $\boxed{\textbf{TRUE}}$:

   $\boxed{\text{A}}$ A drawback of declarative programming languages is that, in order to solve a given problem, they require more lines of code than other more conventional programming languages like Pascal.

   $\boxed{\text{B}}$ Having a higher level, declarative programs are simpler to maintain than the corresponding imperative programs.

   $\boxed{\text{C}}$ There is no declarative and object-oriented programming language.

   $\boxed{\text{D}}$ Declarative language have no practical applications.

31. Which of the following associations between programming languages and features is $\boxed{\textbf{TRUE}}$:

   $\boxed{\text{A}}$ imperative language - inheritance

   $\boxed{\text{B}}$ functional language - polymorphism

   $\boxed{\text{C}}$ logic language - explicit states

   $\boxed{\text{D}}$ imperative language - higher-order

32. Which of the following associations is $\boxed{\textbf{WRONG}:}$

   $\boxed{\text{A}}$ Imperative paradigm ⇔ the execution proceeds as a sequence of states which are necessary to obtain the solution

   $\boxed{\text{B}}$ Declarative paradigm ⇔ the instructions are formulas of some logic system

   $\boxed{\text{C}}$ Imperative paradigm ⇔ the following principle holds: "PROGRAM = SPECIFICATION OF A PROBLEM".

   $\boxed{\text{D}}$ Declarative paradigm ⇔ the following principle holds: "PROGRAM = LOGIC + CONTROL".

33. Which of the following claims is $\boxed{\textbf{TRUE}}$:

   $\boxed{\text{A}}$ Concurrent programming starts in the late 50s with the introduction of the notion of interruption.

   $\boxed{\text{B}}$ Imperative programming closes the gap between specification and programming.

   $\boxed{\text{C}}$ The basic concept of declarative programming is that of state, which is given by the values associated to program variables during the execution.

   $\boxed{\text{D}}$ The programming language Java has no notion of inheritance.