Examen de la convocatoria de Septiembre de 2009

Bloque I. Preguntas cortas LPOO (Java, C++ y C#) 5 puntos

Nombre:_____

DNI: _____ **Titulación:** _____

Se debe tener un mínimo del 40% de la puntuación de este bloque para poder aprobar el examen.

1. ¿Qué diferencia existe entre los paquetes de Java y los espacios de nombres en C#?

2. Implementa el código C# equivalente a la siguiente declaración Java definiendo la propiedad `Edad` que sustituya al método `getEdad`. Nota: puedes suponer la existencia de la clase `System.DateTime` en C# con la misma funcionalidad que la clase `java.util.Date`.

```
import java.util.Date;

public class Persona{
   private Date fechaNacimiento;
   public Persona(Date f){
       fechaNacimiento = f;
   }
   public int getEdad(){
      return new Date().getYear()-fechaNacimiento.getYear();
   }
}
```

3. ¿En qué caso sería válida la siguiente declaración C++? Justifica la respuesta.

```
Cuenta* cuentas = new Cuenta[10];
```

4. En los lenguajes estudiados en la asignatura es posible sobrecargar los constructores de la clase. Explica en cada caso cómo es posible reutilizar el código de dichos constructores.

5. ¿Cuál sería el resultado de la compilación y ejecución del siguiente código C++? Justifica la respuesta.

```
Cuenta cta1, cta2;
cta1 = cta2;
if (cta1 == cta2)
     cout<<”Cuentas iguales”<<endl;
else
     cout<<”Cuentas distintas”<<endl;
```

6. En los lenguajes estudiados en la asignatura, ¿es posible indicar en la declaración de un método que no va a modificar ningún atributo del objeto receptor?

7. En los lenguajes estudiados en la asignatura, si declaramos un bloque `try-catch` que envuelve un fragmento de código que puede lanzar una excepción, ¿cuántos manejadores `catch` podrían dar tratamiento a esa excepción? Justifica la respuesta.

8. ¿Cuál es el significado y cómo se invocaría al siguiente operador C#?

```
public static implicit operator String (Persona p)
{
    return p.Dni;
}
```

9. Explica los tipos de redefiniciones de métodos en C# indicando claramente la diferencia entre ellos.

10. En los lenguajes estudiados en la asignatura, ¿puede una clase genérica ser parametrizada con una clase abstracta? Justifica la respuesta.

11. Completa el código C# del método `main` para que aplicando los métodos de las interfaces la ejecución del programa de como salida la cadena "`1 2`".

```
interface I1{
   int met();
}
interface I2{
   int met();
}
class A : I1, I2{
    int I1.met(){return 1;}
    int I2.met(){return 2;}
    public static void main (){
       A oa = new A();

         //completar

    }
}
```

12. El método C# `GetGrupo` de la clase `Empresa` recorre la lista de empleados (atributo `plantilla`) de la empresa y devuelve un iterador sobre los empleados que cumplen una condición que se le pasa como parámetro. Por ejemplo, una de estas condiciones podría ser que la antigüedad sea mayor de 10 años. Completa la implementación del método que se da a continuación y define todos los tipos adicionales que sean necesarios.

```
public IEnumerable<Empleado> GetGrupo( //completar)
{
    foreach (Empleado empleado in plantilla)
    {









    }
}
```

13. ¿Existe algún problema en el siguiente código C++? En el caso de que lo haya indica la manera de solucionarlo. Justifica la respuesta.

```
class PersonaUMU{
  private:
     string nombre;
  public:
     PersonaUMU(string s);
     string getNombre();
};

class Alumno: public PersonaUMU{…};

class Profesor: public PersonaUMU{…};

class ProfesorAyudante: public Alumno,
                        public Profesor{…};
```

```
PersonaUMU* p;
ProfesorAyudante* ayu;
ayu = new … ;

p = ayu;

p->getNombre();
```

14. Indica si el siguiente enunciado es verdadero o falso, justificando la respuesta: “En los lenguajes estudiados en la asignatura es posible ampliar el conjunto de operaciones aplicables sobre los tipos genéricos, incluida la creación de objetos, siempre y cuando se restrinja la genericidad convenientemente”.

15. Escribe el código C++ equivalente al siguiente código Java:

```
package util;

public class BancoUtil {

  public int procesaBalance(Cuenta cuenta) {

    if (cuenta instanceof CuentaRemunerada)
       return ((CuentaRemunerada)cuenta).getBalance();
    else
       return -1;
  }
}
```