SIMULACRO DE EXAMEN - PROGRAMACION

Unidades 6, 7 y 8: clases avanzadas, colecciones y ficheros

Nombre:		**Fecha:**	
Tiempo recomendado:	60-75 minutos	**Puntuacion:**	10 puntos

Instrucciones: responde en los espacios habilitados. En las preguntas tipo test marca una unica opcion. No uses ordenador ni IDE. El objetivo es simular una prueba real.

1. Colecciones, extraccion de datos e iterador. (1,25 puntos)

Dado el siguiente codigo que recorre un conjunto de pedidos y extrae el nombre del cliente:

```
Set<String> conjuntoPedidos = new LinkedHashSet<>();
conjuntoPedidos.add("Urgente - Ana - 120");
conjuntoPedidos.add("Normal - Luis - 80");
conjuntoPedidos.add("Urgente - Marta - 200");
conjuntoPedidos.add("Normal - Pedro - 50");
conjuntoPedidos.add("Urgente - Sofia - 150");

ArrayList<String> listaClientes = new ArrayList<>();
String cliente;

for (String entrada : conjuntoPedidos) {
    cliente = entrada.substring(
            entrada.indexOf(" - ") + 3,
            entrada.lastIndexOf(" - ")
    ).trim();
    listaClientes.add(cliente);
}
```

a) Indica el contenido final de **listaClientes**.

b) Escribe el codigo necesario para eliminar de **conjuntoPedidos** todos los elementos que empiecen por "Urgente". Debe hacerse sin provocar error al modificar la coleccion durante el recorrido.

c) Indica el contenido final de **conjuntoPedidos** despues de la eliminacion.

2. Listas y metodos genericos. (1,25 puntos)

```
ArrayList<String> productos = new ArrayList<>();
productos.add("anillo plata");
productos.add("colgante ambar");
productos.add("pulsera cuero");
productos.add("pendiente oro");
```

a) Escribe un metodo generico llamado **mostrarLista** que reciba un **ArrayList<T>** y muestre sus elementos por pantalla.

b) Escribe el codigo para convertir todos los elementos de **productos** a mayusculas modificando la propia lista.

c) Indica si el siguiente metodo es correcto o incorrecto y justifica tu respuesta:

```
public static void imprimir(T dato) {
    System.out.println(dato);
}
```

3. Conjuntos, operaciones y ordenacion. (1,25 puntos)

```
Set<Integer> conjuntoA = new HashSet<>();
conjuntoA.add(8);
conjuntoA.add(3);
conjuntoA.add(15);
conjuntoA.add(10);
conjuntoA.add(5);

Set<Integer> conjuntoB = new HashSet<>();
conjuntoB.add(5);
conjuntoB.add(10);
conjuntoB.add(12);
conjuntoB.add(15);
conjuntoB.add(20);
```

a) Declara **conjuntoResultado** para que contenga los elementos de **conjuntoA** ordenados de menor a mayor.

b) Indica el contenido final de **conjuntoResultado** despues de ejecutar:

```
conjuntoResultado.retainAll(conjuntoB);
conjuntoResultado.add(7);
conjuntoResultado.addAll(conjuntoB);
conjuntoResultado.remove(15);
```

c) Explica brevemente por que no aparecen elementos repetidos.

4. Ficheros: orden de ejecucion. (1 punto)

Considerando el siguiente fragmento de codigo desordenado, indica el orden correcto para leer todas las lineas de un fichero llamado **alumnos.txt**.

```
1. while ((linea = br.readLine()) != null) {
       System.out.println(linea);
   }

2. File archivo = new File(System.getProperty("user.dir")
        + File.separator + "recursos"
        + File.separator + "alumnos.txt");

3. String linea;

4. BufferedReader br = new BufferedReader(fr);

5. FileReader fr = new FileReader(archivo);

6. br.close();
```

Orden correcto:

b) Indica un import valido para usar las clases anteriores.

c) Explica que ventaja tiene usar **File.separator**.

5. Herencia y modificadores. (1,5 puntos)

Una academia crea la siguiente jerarquia:

Curso

CursoOnline CursoPresencial

Atributo	Tipo	Descripcion
NOMBRE_ACADEMIA	String	Vale "Academia Aguadulce". Nunca cambia y es compartido por todos.
MAX_ALUMNOS	int	Vale 30. Nunca cambia y es compartido por todos.
totalCursos	int	Contador compartido por todos, pero puede cambiar.
idCurso	int	Identificador exclusivo de cada objeto; no cambia tras asignarse.
titulo	String	Exclusivo de cada objeto y puede modificarse.
horas	int	Accesible desde subclases y puede cambiar.
nivel	String	Accesible desde subclases, pero no puede cambiar una vez asignado.

a) Declara todos los atributos de la clase **Curso** con los modificadores correctos.

b) Indica que atributos son accesibles directamente desde **CursoOnline** sin usar getters y por que.

c) Que ocurriria si intentamos modificar **NOMBRE_ACADEMIA** desde un metodo?

6. Clase abstracta, interfaces y polimorfismo. (1,25 puntos)

```
public abstract class Dispositivo {
    protected String modelo;
    public Dispositivo(String modelo) { this.modelo = modelo; }
    public String getModelo() { return modelo; }
    public abstract void iniciar();
    public void diagnosticar() {
        System.out.println("Diagnosticando " + getModelo());
    }
}

public interface Encendible { void encender(); }
public interface Recargable { void recargar(); }

public class Portatil extends Dispositivo implements Encendible, Recargable {
    public Portatil() { super("Portatil Lenovo"); }
    public void iniciar() { System.out.println("Iniciando portatil"); }
    public void encender() { System.out.println("Encendiendo portatil"); }
    public void recargar() { System.out.println("Recargando portatil"); }
}

public class Impresora extends Dispositivo implements Encendible {
    public Impresora() { super("Impresora HP"); }
    public void iniciar() { System.out.println("Preparando impresora"); }
    public void encender() { System.out.println("Encendiendo impresora"); }
    public void imprimir() { System.out.println("Imprimiendo documento"); }
}

Dispositivo miPortatil = new Portatil();
Dispositivo miImpresora = new Impresora();
Portatil portatilEspecial = new Portatil();
```

a) Si ejecutamos **miPortatil.recargar();**, que ocurrira? Explica por que.

b) Si eliminamos **iniciar()** de **Impresora**, que sucederia?

c) Que metodos puede llamar directamente **portatilEspecial**?

d) Dibuja o describe el esquema de relaciones entre clase abstracta, clases hijas e interfaces.

7. Preguntas tipo test. Marca una unica opcion. (1,25 puntos)

7.1. Que paquete proporciona principalmente las clases clasicas para E/S en Java?

a) java.util.io
b) java.io
c) java.files
d) java.reader

7.2. Que hace deleteOnExit()?

a) Borra inmediatamente el fichero
b) Marca el fichero para borrarlo al finalizar la JVM
c) Crea una copia de seguridad
d) Borra solo directorios vacios

7.3. Si una clase implementa FilenameFilter, que metodo debe implementar?

a) filter()
b) compareTo()
c) accept()
d) readLine()

7.4. Que estructura no admite elementos duplicados?

a) ArrayList
b) LinkedList
c) Set
d) BufferedReader

7.5. Donde se declara en un metodo generico?

a) Despues del nombre del metodo
b) Antes del tipo de retorno
c) Dentro del main
d) Al final de la clase

---

8. Serializacion y excepciones. (1,25 puntos)

Completa o responde brevemente:

a) Si queremos guardar objetos de tipo **Alumno** con **ObjectOutputStream**, que interfaz debe implementar la clase?

---

b) Completa la cabecera:

```
public class Alumno _____________ {
    private String nombre;
    private int edad;
}
```

---

c) Escribe una estructura try-with-resources para crear un **ObjectOutputStream** asociado a "alumnos.dat".

d) Indica una excepcion que podria aparecer al trabajar con ficheros.

Hoja de control del alumno

Usa esta pagina al terminar para anotar tus tiempos y sensaciones antes de corregir.

Inicio	Fin	Tiempo total	Sensacion general
Ejercicio mas facil		Ejercicio mas dificil	
Dudas principales		Nota estimada	

Importante: no mires apuntes durante la prueba. Cuando termines, escribe tus respuestas en el chat o sube fotos del examen y lo corregimos como simulacro real.