

EXAMEN PDOO : TEORÍA 2 ~ [2019 - 2020]

TIPO DE EXAMEN : 1

Pregunta 1 - [10 puntos]: Dados los siguientes ficheros

```
00: //FICHERO HIJA.JAVA
01: package paqueteA;
02:
03: class Padre{
04:     protected void protegido(){
05:         System.out.println("Protegido Padre");
06:     }
07:     void metodo(){
08:         System.out.println("Metodo Padre");
09:     }
10:     public void procesa(){
11:         System.out.println("Procesando en el padre...");
12:     }
13:     public void ejecutarTarea(){
14:         procesa();
15:         System.out.println("Fin de la tarea en el padre");
16:     }
17: }
18:
19: public class Hija extends Padre{
20:     void test(Padre p){
21:         p.protegido();
22:         p.metodo();
23:     }
24: }
```

```
00: //FICHERO NIETA.JAVA
01: package subpaquete.paqueteA;
02: import paqueteA.Hija;
03:
04: public class Nieta extends Hija{
05:     void test(Hija p){
06:         p.protegido();
07:         p.metodo();
08:     }
09:     void test2(Nieta p){
10:         p.protegido();
11:         p.metodo();
12:     }
13:     @Override
14:     public void procesa(){
15:         System.out.println("Procesando en el nieto...");
16:     }
17:     public void tareaNieta(){
18:         System.out.println("Tarea en el nieto");
19:     }
20: }
```

```

00: //FICHERO PRINCIPAL.JAVA
01: import java.util.ArrayList;
02: import paqueteA.Hija;
03: import subpaquete.paqueteA.Nieta;
04:
05: interface MyInterface{
06:     public void ejecutarTarea();
07: }
08:
09: public class Principal{
10:     public static void main(String[] args){
11:         Nieta n=new Nieta();
12:         n.ejecutarTarea();
13:         ((Hija) n).ejecutarTarea();
14:
15:         Hija h=new Nieta();
16:         MyInterface interf=h;
17:
18:         h.tareaNieta();
19:
20:         ArrayList<Integer> array = (ArrayList<Integer>) (Object) h;
21:         ArrayList<Hija> array2 = new ArrayList<Nieta>();
22:     }
23: }

```

Encontrar los errores asociados a la visibilidad/especificadores de acceso. Indicar:

- fichero
- número de línea
- explicación de la causa

Pregunta 2 - [10 puntos]: Dados los siguientes ficheros

```
00: //FICHERO B.JAVA
01: package paqueteA;
02:
03: class A{
04:     protected void protegido(){
05:         System.out.println("Protegido A");
06:     }
07:     void metodo(){
08:         System.out.println("Metodo A");
09:     }
10:     public void procesa(){
11:         System.out.println("Procesando en A...");
12:     }
13:     public void ejecutarTarea(){
14:         procesa();
15:         System.out.println("Fin de la tarea en A");
16:     }
17: }
18:
19: public class B extends A{
20:     void test(A p){
21:         p.protegido();
22:         p.metodo();
23:     }
24: }
```

```
00: //FICHERO C.JAVA
01: package subpaquete.paqueteA;
02: import paqueteA.B;
03:
04: public class C extends B{
05:     void test2(C p){
06:         p.protegido();
07:         p.metodo();
08:     }
09:     @Override
10:     public void procesa(){
11:         System.out.println("Procesando en C...");
12:     }
13:     public void tareaC(){
14:         System.out.println("Tarea en C");
15:     }
16:     void test(B p){
17:         p.protegido();
18:         p.metodo();
19:     }
20: }
```

```

00: //FICHERO PRINCIPAL.JAVA
01: import java.util.ArrayList;
02: import paqueteA.B;
03: import subpaquete.paqueteA.C;
04:
05: interface MyInterface{
06:     public void ejecutarTarea();
07: }
08:
09: public class Principal{
10:     public static void main(String[] args){
11:         ArrayList<Integer> array = (ArrayList<Integer>) (Object) h;
12:         ArrayList<Hija> array2 = new ArrayList<Nieta>();
13:
14:         C n=new C();
15:         n.ejecutarTarea();
16:         ((B) n).ejecutarTarea();
17:
18:         B h=new C();
19:         MyInterface interf=h;
20:
21:         h.tareaC();
22:     }
23: }

```

Encontrar los errores asociados a compatibilidad entre tipos y cuestiones relativas a operaciones de *casting*. Indicar:

- fichero
- número de línea
- explicación de la causa indicando si el error se produce en tiempo de compilación o ejecución.

Pregunta 3 - [8 puntos]: Dados los siguientes ficheros

```
00: //FICHERO B.JAVA
01: package paqueteA;
02:
03: class A{
04:     protected void protegido(){
05:         System.out.println("Protegido A");
06:     }
07:     void metodo(){
08:         System.out.println("Metodo A");
09:     }
10:     public void procesa(){
11:         System.out.println("Procesando en A...");
12:     }
13:     public void ejecutarTarea(){
14:         procesa();
15:         System.out.println("Fin de la tarea en A");
16:     }
17: }
18:
19: public class B extends A{
20:     void test(A p){
21:         p.protegido();
22:         p.metodo();
23:     }
24: }
```

```
00: //FICHERO C.JAVA
01: package subpaquete.paqueteA;
02: import paqueteA.B;
03:
04: public class C extends B{
05:     void test2(C p){
06:         p.protegido();
07:         p.metodo();
08:     }
09:     @Override
10:     public void procesa(){
11:         System.out.println("Procesando en C...");
12:     }
13:     public void tareaC(){
14:         System.out.println("Tarea en C");
15:     }
16:     void test(B p){
17:         p.protegido();
18:         p.metodo();
19:     }
20: }
```

```

00: //FICHERO PRINCIPAL.JAVA
01: import java.util.ArrayList;
02: import paqueteA.B;
03: import subpaquete.paqueteA.C;
04:
05: interface MyInterface{
06:     public void ejecutarTarea();
07: }
08:
09: public class Principal{
10:     public static void main(String[] args){
11:         ArrayList<Integer> array = (ArrayList<Integer>) (Object) h;
12:         ArrayList<Hija> array2 = new ArrayList<Nieta>();
13:
14:         C n=new C();
15:         n.ejecutarTarea();
16:         ((B) n).ejecutarTarea();
17:
18:         B h=new C();
19:         MyInterface interf=h;
20:
21:         h.tareaC();
22:     }
23: }

```

Indicar la salida que se produce en la consola al ejecutar el programa principal ignorando las líneas con errores:

Pregunta 4 - [7 puntos]: Dados el siguiente código

```
package deepspace;
import java.util.ArrayList;

class SpaceAssetsSet{
    private ArrayList<Object> assets=new ArrayList();

    public void add(Object o) {assets.add(o);}
    public ArrayList<Object> getElements() {return assets;}
}
```

```
public class Examen{
    public static void main(String[] args){
        SpaceAssetsSet sa=new SpaceAssetsSet();
        //Se añaden elementos
        ArrayList<Object> assets=sa.getElements();

        float value=0.0f;
        for(Object a:assets){
            if(a instanceof Weapon){
                value += ((Weapon) a).power()*1.5f;
            }
            else{
                if(a instanceof ShieldBooster){
                    value += ((ShieldBooster) a).getBoost()*1.3f;
                }
                else{
                    //Asumimos que tiene que ser una SpaceStation
                    SpaceStation ss=(SpaceStation) a;
                    for(Weapon w:ss.getWeapons()){
                        value += w.power()*1.5f;
                    }
                    for(ShieldBooster s:ss.getShieldBoosters()){
                        value += s.getBoost()*1.3f;
                    }
                    Hangar h=ss.getHangar();
                    if(h!=null){
                        for(Weapon w:h.getWeapons()){
                            value += w.power()*1.5f;
                        }
                        for(ShieldBooster s:h.getShieldBoosters()){
                            value += s.getBoost()*1.3f;
                        }
                    }
                }
            }
        }
    }
}
```

¿Qué problemas encuentras asociados al diseño software proporcionado? Proponer una estrategia de rediseño indicando cómo quedaría el código una vez aplicada. Se puede utilizar código fuente para contestar la pregunta.

