Examen de **Tecnología de Programación** **9 de Septiembre de 2015**
Grado en Ingeniería Informática.
Escuela de Ingeniería y Arquitectura.
Universidad de Zaragoza

Duración total del examen: ***2 horas 30 minutos***

NOTA RECORDATORIA: La evaluación final consta de las siguientes pruebas:

- Prueba escrita → 60% de la nota.
- Prácticas, la máxima nota entre las prácticas entregadas y defendidas durante el curso y el examen de prácticas → 40% de la nota.

Ejercicio 1 [2 puntos]

A continuación verás dos programas realizados en el lenguaje Java. Por cada uno de los programas podrá haber más de un fichero: el nombre del fichero aparecerá encima de su correspondiente código. Asume que todos los ficheros están en la misma carpeta. De cada uno de los programas, responde a las siguientes preguntas:

- ¿Compilaría correctamente o daría algún error? Ignora errores tipográficos.
- En caso de que en tu respuesta anterior indiques que no compila, ¿por qué no compila?
- En caso de que en tu primera respuesta indiques que compila, ¿qué sacaría el programa por pantalla? No es necesario que lo justifiques.

Programa A

Foo.java

```
public class Foo<T>
{
  T v;

  Foo(T v)
  { this.v = v; }

  public T value()
  { return v+1; }
}
```

Bar.java

```
public class Bar
    extends Foo<Integer>
{
   public Bar(Integer v)
   { super(v); }

  public Integer value()
  { return
      super.value()-1; }
}
```

Pet.java

```
public class Pet
{
   private int demore;
   public void ecan(Foo<Integer> sion)
   {  demore += sion.value(); }
   public int value()
   {  return demore; }
   public static void main(String[] args){
      Pet pet = new Pet();
      pet.ecan(new Bar(21));
      pet.ecan(new Foo<Integer>(20));
      System.out.println(pet.value());
   }}
```

Programa B

Foo.java

```
public class Foo
{
 public String
           ricada()
 {
    return "Dar";
 }
}
```

Bar.java

```
public class Bar
            extends Foo
{
  String tillery;

  public Bar(String t)
 { this.tillery =
              (t+"e"); }

  public String ricada()
  { return super.ricada()
          +tillery+"r"; }
}
```

Pet.java

```
public class Pet extends Bar
{
  public Pet(String strong)
   {  super(strong+" Vad"); }

  public static void main(String[] args)
  {
     Bar bar = new Bar("k Matt");
     System.out.println(bar.ricada());
     Pet pet = new Pet("th");
     System.out.println(pet.ricada());
   }
}
```

Duración total del examen: ***2 horas 30 minutos***

Ejercicio 2 [3 puntos]

La **herencia** es un mecanismo de ciertos lenguajes de programación que permite establecer relaciones entre diferentes tipos de datos. Dentro de una jerarquía de herencia, la **asociación** de métodos (o funciones) relacionadas con los tipos de datos involucrados puede ser **estática o dinámica**. Sobre este tema, realiza las siguientes tareas.

(a) Explica las **diferencias** entre ambos modos de asociación.

(b) Indica cuál es el modo de **asociación por defecto** en los siguientes lenguajes: Java, C++, Haskell.

(c) Pon un **ejemplo de código** con herencia en un lenguaje orientado a objetos a tu elección, que presente diferente comportamiento (muestre cosas diferentes por pantalla) dependiendo de si la asociación es estática o dinámica.

(d) Escribe una **traza de ejecución** del código anterior (es decir, lo que sacaría el programa por pantalla) suponiendo que la asociación es estática. Repite lo mismo suponiendo que la asociación es dinámica. Evidentemente, ambas trazas de ejecución deberían ser diferentes.

(e) Explica, en base a tu respuesta en (a), las diferencias entre ambas trazas de ejecución planteadas en (d).

Duración total del examen: ***2 horas 30 minutos***

Ejercicio 3 [2 puntos]

En matemáticas un número de ***Harshad*** es un natural que es divisible por la suma de sus dígitos (operando en una base dada). Estos números fueron definidos por D. R. Kaprekar y la palabra Harshad proviene del sánscrito y significa “gran alegría”.

Ejemplos:

- El número 18 es un número Harshad en base 10, ya que la suma de sus dígitos es 9 (1+8=9), y 18 es divisible por 9.
- El número 19 no es un número de Harshad, ya que la suma de sus dígitos es 10 (1+9=10), y 19 no es divisible por 10.
- El número 1729 es un número de Harshad, ya que la suma de sus dígitos es 19, y 1729=19*91.

Se pide:

(a) **Desarrolla** en Haskell la función **`digitos`**, que dado un número natural (de precisión arbitraria) devuelva una lista con todos sus dígitos en cualquier orden (aunque es recomendable que estén ordenados de menos a más significativo)

```
> digitos 547
> [7,4,5]
```

(b) Utilizando la función del apartado (a), desarrolla en Haskell la función **`harshad`** que (sin ningún parámetro) devuelva una lista infinita con todos los números de Harshad (en base 10) ordenados de menor a mayor.

(c) Utilizando la función **`harshad`**, desarrolla en Haskell la función **`harshadBounded a b`**, que devuelva una lista con todos los números de Harshad entre los números *a* y *b* (ambos inclusive). Asegura que la lista devuelta puede imprimirse por pantalla y no es infinita.

```
> harshadBounded 10 60
> [10,12,18,20,21,24,27,30,36,40,42,45,48,50,54,60]
```

Duración total del examen: ***2 horas 30 minutos***

Ejercicio 4 [3 puntos]

La **compañía de telecomunicaciones** *TamTamPhone* está renovando su software de gestión de clientes. Esta compañía ofrece servicios de **televisión**, **internet**, y **telefonía móvil**, y es famosa por sus grandes ofertas. Dado un conjunto de servicios solicitados por un cliente, la compañía pretende implementar un sistema que busque la mejor oferta para dicho cliente.

Implementa el conjunto de clases/interfaces necesarios para modelar los siguientes objetos del sistema:

- **Servicios**, que pueden ser televisión, internet o móvil. Cada servicio tiene un precio determinado, que puede variar según sus características:
  - Televisión: numero de canales
  - Internet: velocidad
  - Móvil: cuota de datos
- **Petición**, que puede incluir o no los servicios de internet, televisión y/o móvil (como máximo uno de cada tipo).
- **Oferta**, que puede incluir o no los servicios de internet, televisión y/o móvil (como máximo uno de cada tipo), y tiene un precio menor que el conjunto de los servicios que incluye por separado.

Implementa un método que a partir de una Petición y una lista de Ofertas, nos devuelva las ofertas que incluyan AL MENOS todos los servicios que pide el cliente y que cuesten menos que los servicios solicitados por separado (si existen):

```
ArrayList<Oferta> ofertasInteresantes(Peticion peticion, ArrayList<Oferta> todasLasOfertas)
```

No hay problema en que una oferta incluya algún servicio no solicitado, mientras cueste menos.