PRUEBA 1 PROGRAMACIÓN **Noviembre 2008** **INGENIERÍA INFORMÁTICA** **Leganés**	Universidad Carlos III de Madrid

LEA <u>ATENTAMENTE</u> ESTAS INSTRUCCIONES ANTES DE COMENZAR LA PRUEBA:

- **Rellene todas las hojas a <u>bolígrafo,</u> tanto los datos personales como las respuestas**
- **No utilice lápiz ni bolígrafo rojo**
- **No olvide rellenar el NIA y el grupo <u>real</u> al que pertenece**
- **El tiempo máximo de realización es de 1 hora**
- **El único material permitido sobre la mesa es la hoja de test y un bolígrafo**
- **Utilice exclusivamente esta hoja de test para las respuestas, use las caras posteriores para contestar si lo necesita. <u>No se recogerá</u> ninguna otra hoja adicional.**

<u>NO PASE DE ESTA HOJA</u>, hasta que se le indique

Apellidos	***Nombre***	
Firma	***NIA***	***Grupo***

PARTE 1: CUESTIONES

Pregunta 1 (1 Punto).- Indicar si la siguiente afirmación es cierta, y **explicar** brevemente por qué.
"Utilizando & en lugar de && siempre obtendremos el mismo resultado en una expresión booleana"

Verdadero. Tanto & como && realizan la operación lógica AND, la diferencia entre ambos está en que & evalúa los dos términos de la expresión, mientras que && en el caso de que el primero sea falso (y por lo tanto el resultado será falso independientemente del valor del segundo) no evalúa el segundo. Por lo tanto el resultado de la operación lógica no varía, aunque el uso de uno u otro puede tener efectos colaterales distintos como por ejemplo:

```
int a=0;
boolean b = a>0 & 5/a>1;
```

Daría un error de ejecución que no se produciría en el caso de usar &&

Pregunta 2 (1 Punto).- Indicar si la siguiente afirmación es cierta, y **explicar** brevemente por qué.
"Si declaramos una variable de un tipo básico y no le damos valor, Java le da un valor por defecto (0 para los tipos numéricos, false para boolean y el carácter vacío para char)"

Falso. Java no da valores por defecto a las variables de tipos básicos, por lo que es obligatorio darles un valor antes de usarlas. Estos valores por defecto sí se dan a los distintos elementos de un array.

Pregunta 3 (1 Punto).- Indicar si la siguiente afirmación es cierta, y **explicar** brevemente por qué.
"Podemos mezclar variables de distintos tipos numéricos en una expresión aritmética y Java decidirá automáticamente el tipo del resultado"
Verdadero. En una expresión aritmética se pueden mezclar todos los tipos numéricos, e incluso `char`, el tipo del resultado será el del operando de mayor rango, excepto en el caso en el que se opere exclusivamente con `char`, `byte` o `short` en cuyo caso el resultado será de tipo `int`

Pregunta 4 (1 Punto).- Encontrar y **explicar** los 3 errores de compilación que aparecen en el siguiente código Java. ¿Cómo los resolvería?

```
public class ProblemaCuatro {
	public static void main(String[] args) {
	int a, b, c;
	short d=125,e = 4, f;
	char g='g';
	float h = 3, i, a=3.5F;
	double j=3.2F;
	a=4;
	c= a+b;
	e=45;
	f=e+d;
	System.out.println("El valor de f es: "+(e+d));
	System.out.println("La letra siguiente a la g es: "+(++g));

	}

}
```

Primer error: se declara dos veces una variable de nombre a, una como `int` y otra como `float`. Habría que cambiar el nombre a alguna de las dos.
Segundo error: Se hace `c = a+b` sin haber dado antes valor a `a` ni a `b`. Habría que haberles dado un valor antes.
Tercer error: al hacer `e + d` el resultado es automáticamente de tipo `int`, por lo que no se puede almacenar en la variable `f` que es `short`. Para que funcionara habría que haber hecho un casting: `f = (short) (e+d)`

Pregunta 5 (1 Punto).- <u>Explicar</u> el resultado de compilar y ejecutar el siguiente programa:

```
public class ProblemaCinco {
	public static void main(String[] args) {
		for(int i=0; i<=2; i++) {
			for(int j=0; j<2; j++) {
				System.out.println(j*i);
			}
		}
	}

}
```

El programa tiene dos bucles for anidados, el primero se ejecuta entre 0 y 2 y el segundo entre 0 y 1, cada vez que se ejecuta el segundo bucle se imprime el producto de los dos índices. Por lo tanto el resultado será:
i=0, j=0 →imprime 0
i=0, j=1 →imprime 0
i=1, j=0 →imprime 0
i=1, j=1 →imprime 1
i=2, j=0 →imprime 0
i=2, j=1 →imprime 2

Pregunta 6 (1 Punto).- Dada las siguientes declaraciones, **explicar** cuáles son correctas y cuáles no. En las incorrectas, ¿hay alguna forma de corregirlas sin cambiar el tipo de las variables? En caso afirmativo, diga cómo se haría.

```
a) int a= 300; byte b = a;
b) int a1 = 90; char b1=a1;
c) float a2= 4; double b2=a2/0;
d) char a3 = 'g', b3 = a3+1;
e) long a4 = 5; int b4=a4/3;
```

a) Incorrecta, intentamos asignar un dato de tipo `int` a una variable de tipo `byte`. Se resolvería con un casting: `byte b = (byte) a`, aunque al superar 300 el rango de `byte` el resultado almacenado en b no será el deseado.
b) Incorrecta, intentamos asignar un dato de tipo `int` a una variable de tipo `char`. Se resolvería también con un casting: `char b1 = (char) a1`
c) Correcta, asignamos un `int` a un `float` y luego a un `double` que valdrá `Infinity`
d) Incorrecta, al sumar 1 a un `char`, se convierte automáticamente en un `int`, y pasa lo mismo que en b). Se resuelve con un casting: `b3 = (char) (a3+1)`
e) Incorrecta, asignamos un `long` a una variable de tipo `int`. Se resuelve con un casting: `int b4= (int) a4/3`

PARTE 2: PROBLEMAS

Problema 1 (2 Puntos).- En la asignatura de programación del grado de ingeniería informática hay 5 grupos de alumnos, que tienen respectivamente 35, 32, 31, 37 y 39 alumnos. Crear **un** array de `String` para representar esta estructura de forma que cada posición del array pueda contener el nombre y apellidos de un alumno. Crear un alumno en cada uno de los grupos.

```
String [][] alumnos = new String [5][];
alumnos [0] = new String [35];
alumnos [1] = new String [32];
alumnos [2] = new String [31];
alumnos [3] = new String [37];
alumnos [4] = new String [39];

alumnos[0][0] = "Pepe Sánchez";
alumnos[1][0] = "Luis Sánchez";
alumnos[2][0] = "Marisa Molina";
alumnos[3][0] = "Juana Gómez";
alumnos[4][0] = "Eladio Pérez";
```

Problema 2 (2 Puntos).- Crear un array de 2000 posiciones de tipo `double` y rellenarlo de la siguiente manera:

- Las casillas pares deberán tener como valor el triple de su índice, es decir la 0, contendrá 0, la 2 valdrá 6, la 4 valdrá 12, y así sucesivamente.
- Las casillas impares deberán tener como valor el índice en negativo, es decir, la 1 contendrá -1, la 3 valdrá -3 y así sucesivamente.

```
double array [] = new double [2000];
// recorremos el array con un bluce for
for (int i=0; i < array.length; i++){
      if (i%2==0) array[i]=3*i;
      else array[i]=-i;
      }
```