Material para
a Formación Profesional inicial

A01. Uso de obxectos predefinidos

Familia profesional	IFC	Informática e comunicacións
Ciclo formativo	CSIFC02 CSIFC03	Desenvolvemento de aplicacións multiplataforma Desenvolvemento de aplicacións web
Grao		Superior
Módulo profesional	MP0485	Programación
Unidade didáctica	UD04	Conceptos avanzados de POO
Actividade	**A01**	**Uso de obxectos predefinidos**
Autores		Silvia Framiñán Fondevila Marta Rey López
Nome do arquivo		CSIF02_MP0485_V000401_UD04_A01_Obxectos_predefinidos.odt

Índice

1. Ficha técnica

Contexto da actividade

Módulo	Duración	Unidade didáctica.	Sesións 60´	Actividades	Sesións 60´
MP0485. Programación.	240	UD01. Identificación dos elementos dun programa informático.	24	A01. Metodoloxía da programación.	5
				A02. Introdución á linguaxe de programación Java.	7
				A03. Elementos básicos dun programa Java.	18
		UD02. Uso de estruturas de control.	20	A01. Estruturas de control selectivas.	8
				A02. Estruturas de control repetitivas.	12
		UD03. Introdución á POO.	28	A01. Conceptos de POO.	8
				A02. Clases e obxectos.	20
		U04. Conceptos avanzados de POO.	**20**	**A01. Obxectos predefinidos e métodos estáticos.**	**10**
				A02. Constantes e métodos estáticos.	5
				A03. Visibilidade e empaquetaxe.	5
		UD05. Lectura e escritura de información.	34	A01. Fluxos de entrada e saída.	4
				A02. Uso de ficheiros.	15
				A03. Interfaces gráficas de usuario.	15
		UD06. Aplicación das estruturas de almacenamento.	32	A01. Uso de cadeas.	8
				A02. Uso de arrais.	9
				A03.Uso de coleccións e outras estruturas	12
				A04. Manipulación de documentos XML.	8
		UD07. Xerarquías de clases e excepcións.	37	A01. Introdución á herdanza.	10
				A02. Uso avanzado da herdanza.	10
				A03. Interfaces.	10
				A04. Control de código. Excepcións.	7
		UD08. Mantemento da persistencia dos obxectos.	18	A01. Instalación dun SXBDOO e almacenamento básico de obxectos.	8
				A02. Almacenamento e recuperación de obxectos nun SXBDOO e operacións con datos complexos.	10
		UD09. Xestión dos datos almacenados nas bases de datos relacionais.	22	A01. Conexión con sistemas xestores de bases de datos relacionais.	6
				A02. Operacións de lectura nunha base de datos relacional.	7
				A03. Operacións de escritura nunha base de datos relacional e uso de transaccións.	9

NOTA: Esta actividade está vinculada á programación recollida no arquivo CSIFC02_MP0485_V000400_UD04_Conceptos_avanzados_POO.pdf

Título da actividade

Nº	Título	Descrición	Duración

A01	Obxectos predefinidos e métodos estáticos.	Nesta actividade empregaranse algunhas das librarías predefinidas na linguaxe, incorporándoas ao código, instanciando os obxectos e empregando os seu métodos. Ademais aproveitarase o uso dalgunha librería con clases estáticas para introducir e manexo de métodos estáticos.	10

Resultados de aprendizaxe do currículo

Resultados de aprendizaxe do currículo	Completo
▪ RA2: Escribe e proba programas sinxelos, para o que recoñece e aplica os fundamentos da programación orientada a obxectos.	**NON**

Obxectivos didácticos e título e descrición da actividade

Obxectivos específicos		Actividade		Descrición básica	Duración
O1.1	Instanciáronse obxectos a partir de clases predefinidas.	A01	Obxectos predefinidos e métodos estáticos.	Nesta actividade empregaranse algunhas das librarías predefinidas na linguaxe, incorporándoas ao código, instanciando os obxectos e empregando os seu métodos. Ademais aproveitarase o uso dalgunha librería con clases estáticas para introducir e manexo de métodos estáticos.	10
O1.2	Escribíronse chamadas a métodos estáticos.				
O1.3	Incorporáronse e utilizáronse librarías de obxectos.				

Criterios de avaliación

Criterios de avaliación
▪ CA2.3. Instanciáronse obxectos a partir de clases predefinidas.
▪ CA2.5. Escribíronse chamadas a métodos estáticos.
▪ CA2.7. Incorporáronse e utilizáronse librarías de obxectos.

Contidos

Contidos
▪ BC2. Uso de obxectos: – Uso de métodos, de propiedades e de métodos estáticos. – Librarías de obxectos. ▪ BC1. Identificación dos elementos dun programa informático. – Librarías de funcións. – Funcións de usuario.

Actividades de ensino e aprendizaxe e de avaliación, métodos, recursos e instrumentos de avaliación (exemplo)

Que e para que	Como			Con que	Como e con que se valora	
Actividade (título e descrición)	**Profesorado (en termos de tarefas)**	**Alumnado (tarefas)**	**Resultados ou produtos**	**Recursos**	**Instrumentos e procedementos de avaliación**	
A01. Obxectos predefinidos e métodos estáticos. ▪ Nesta actividade empregaranse algunhas das librarías predefinidas na linguaxe, incorporándoas ao código, instanciando os obxectos e empregando os seu métodos. Ademais aproveitarase o uso dalgunha librería con clases estáticas para introducir e manexo de métodos estáticos.	▪ **Tp1.1** - Exposición da clasificación das clases en paquetes e librarías. ▪ **Tp1.2** - Descrición dalgunhas das librarías predefinidas máis coñecidas da linguaxe Java.	▪ **Ta1.1** - Creación dun programa que empregue a clase Math. ▪ **Ta1.2** - Creación dun programa que empregue a clase Random.	▪ Proxecto Java.	▪ Ordenador persoal con conexión a Internet. ▪ Contorno de desenvolvemento NetBeans. ▪ Java SE Development Kit. ▪ Apuntamentos da profesora. ▪ Proxector.		7
		▪ **Ta1.3** - Tarefa de avaliación, combinando: – Táboa de indicadores sobre proxectos Java nos que se instancien obxectos a partir de clases predifinidas. – Táboa de indicadores sobre un proxecto Java no que se escriban chamadas a métodos estáticos. – Táboa de indicadores sobre proxectos Java nos que se incorporen e empreguen librarías de obxectos.	▪ Proxectos Java.		▪ OU 1. Táboa de indicadores sobre proxectos Java nos que se instancien obxectos a partir de clases predifinidas. ▪ OU 2. Táboa de indicadores sobre un proxecto Java no que se escriban chamadas a métodos estáticos. ▪ OU 3. Táboa de indicadores sobre proxectos Java nos que se incorporen e empreguen librarías de obxectos.	3

2. A01. Obxectos predefinidos e métodos estáticos.

2.1 Introdución

Na actividade que nos ocupa aprenderanse os seguintes conceptos e manexo de destrezas sobre a Programación Orientada a Obxectos:

- Explicación e emprego de métodos e atributos estáticos.
- Distribución das clases predefinidas en librarías.
- Clases predefinidas de utilidade.

2.2 Actividade

2.2.1 Métodos e atributos estáticos ou de clase

Un atributo estático ou de clase é aquel que non é específico de cada obxecto, senón que só hai unha copia do mesmo e o seu valor é compartido por todos os obxectos da clase. Existe e pode empregarse aínda que non existan obxectos desa clase.

Un método estático ou de clase é aquel que se invoca sen instanciar un obxecto desa clase. Unha clase estática é aquela que non pode ser instanciada. O emprego dos atributos, métodos ou constantes estáticos farase directamente a partir do nome da clase, por exemplo:

```
double area = Math.PI * Math.pow(radio, 2);
```

2.2.2 Librarías de obxectos

O obxectivo da linguaxe Java é presentar unha sintaxe sinxela, pero tamén ofrecer ao programador un completo conxunto de facilidades para desenvolver aplicacións de propósito moi diverso e sobre un gran número de plataformas diferentes.

Para poder conseguilo, Java ofrece un gran número de librarías de clases e cada unha destas librarías contén un ou varios paquetes. Estes paquetes estrutúranse de forma xerárquica, como un sistema de directorios (que se corresponden cos paquetes) e arquivos (que se corresponden coas clases). Deste xeito, un paquete pode conter clases e outros paquetes. O nome dun paquete levará os nomes de todos os paquetes aos que pertence separados por puntos.

2.2.2.1 Inclusión das librarías no código

Para empregar unha clase incluída dentro dun paquete temos dúas posibilidades:

- Referenciar a clase utilizando o seu nome completo (precedido polos paquetes e subpaquetes aos que pertence):

```
java.util.Date data = new java.util.Date();
```

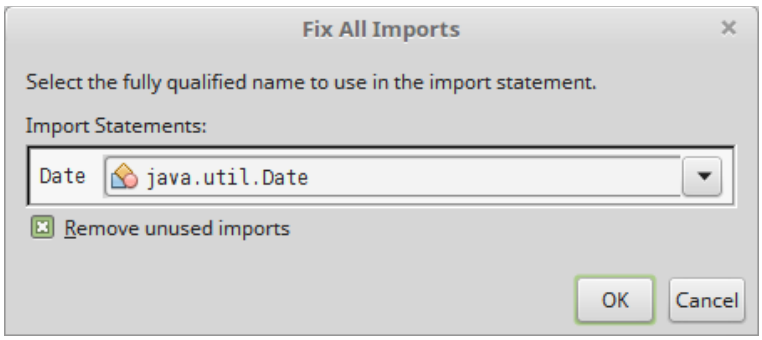
- Importar a clase para non ter que antepoñer o paquete ao que pertence. Para importar as clases incluídas nas librarías debemos empregar a palabra reservada `import`, que debe aparecer antes de calquera declaración de clase no código fonte.

```
import paquete.subpaquete.subsubapquete....clase;
```

Por exemplo, a clase `Date` atópase no paquete `java.util` importarase así:

```
import java.util.Date;
```

No contorno de programación NetBeans, se escribimos código empregando clases sen indicar a importación, indicará un erro subliñando o nome da clase en vermello. Se prememos co botón dereito temos a opción *Fix imports*, que nos permite incluír automaticamente os paquetes necesarios.



2.2.2.2 Acceso á documentación das librarías

Para coñecer cales son as clases que ofrece cada libraría de Java e as súas funcionalidades e métodos, podemos facelo consultando a documentación oficial de Oracle: a especificación da API de Java, que se pode atopar en https://docs.oracle.com/javase/8/docs/api/index.html

2.2.3 Algunhas clases predefinidas en Java

2.2.3.1 O paquete java.lang

Este paquete está incluído en calquera aplicación, polo que non é preciso importalo explicitamente. Contén as clases básicas do sistema, entre outras:

- `System`: proporciona obxectos para entrada e saída e funcionalidades do sistema.
- `Math`: proporciona funcións matemáticas usuais.
- `Object`: a raíz xerárquica da xerarquía de clases, define o comportamento base de todos os obxectos.
- `String` e `StringBuilder`: permite o manexo de cadeas de caracteres.
- Envoltorios de tipos primitivos (`char`, `int`, `double`, etc.).
- `Class`: proporciona información sobre as clases.

java.lang.System

Esta clase proporciona, entre outros, obxectos para a entrada e saída e funcionalidades do sistema.

Proporciona tres variables públicas de clase para a entrada e saída estándar:

- `System.in`: Instancia da clase `InputStream`, implementa a entrada estándar.
- `System.out`: Instancia da clase `PrintStream`, implementa a saída estándar.
- `System.err`: Instancia da clase `PrintStream`, implementa a saída de erro. Funciona de forma similar a `System.out` e emprégase para enviar mensaxes de erro, a un ficheiro ou á consola.

Por exemplo:

```
System.out.println(area);
```

Pode obterse máis información da clase `System` na documentación oficial de Oracle https://docs.oracle.com/javase/8/docs/api/java/lang/System.html

java.lang.Math

Esta clase contén métodos para levar a cabo operacións numéricas, como as exponenciais, logarítmicas raíces cadradas e trigonométricas. É unha clase estática que non pode ser instanciada. A continuación expoñemos algúns dos atributos e métodos máis importantes.

Constante	Descrición
E	O valor de tipo double que é máis próximo ao número *e*, a base dos logaritmos naturais.
PI	O valor de tipo double que é máis próximo ao número *pi*, a razón entre circunferencia dun círculo e o seu diámetro.
Método	**Descrición**
abs()	Valor absoluto do número recibido como parámetro.
ceil()	Aproxima ao enteiro superior máis próximo.
floor()	Aproxima ao enteiro inferior máis próximo.
round()	Aproxima ao enteiro máis próximo.
sin(), cos(), tan(), acos(), asin(), atan()	Implementan as funcións trigonométricas.
max()	Máximo de dous argumentos.
min()	Mínimo de dous argumentos.
exp()	Función exponencial.
log()	Logaritmo natural.
random()	Xera un número aleatorio entre 0 e 1.
pow()	Devolve o primeiro parámetro elevado ao segundo.
sqrt()	Raíz cadrada.

Por exemplo:

```
double area = Math.PI * Math.pow(radio, 2);
areaEnteira = Math.round(area);
```

Pode obterse máis información da clase Math na documentación oficial de Oracle https://docs.oracle.com/javase/8/docs/api/java/lang/Math.html

Realiza a tarefa 3.1. "Creación dun programa que empregue a clase Math", onde se empregará a clase Math para calcular as raíces dun polinomio de segundo grao.

Clases envoltorio

Hai ocasións nas que é preciso traballar cos tipos de datos primitivos (`byte`, `short`, `int`, `long`, `float`, `double`, `char` e `boolean`) coma se fosen obxectos. Para iso existen as clases envoltorio da libraría `java.lang`.

Tipo primitivo	Clase envoltorio
byte	Byte
short	Short
integer	Integer
long	Long
float	Float
double	Double
character	Character
boolean	Boolean

Estas clases permiten traballar con eles coma se fosen obxectos, converter entre tipos e converter valores destes tipos a cadeas de caracteres e viceversa facilmente.

Para "envolver" o tipo primitivo habería que facelo deste xeito:

```
Integer enteiroEnvolto = new Integer(15);
```

E para "desenvolvelo":

```
int enteiroDesenvolto = enteiroEnvolto.intValue();
```

Pode obterse máis información das clases envoltorio na documentación oficial
https://docs.oracle.com/javase/8/docs/api/java/lang/Boolean.html,
https://docs.oracle.com/javase/8/docs/api/java/lang/Byte.html,
https://docs.oracle.com/javase/8/docs/api/java/lang/Character.html,
https://docs.oracle.com/javase/8/docs/api/java/lang/Double.html,
https://docs.oracle.com/javase/8/docs/api/java/lang/Float.html,
https://docs.oracle.com/javase/8/docs/api/java/lang/Integer.html,
https://docs.oracle.com/javase/8/docs/api/java/lang/Long.html,
https://docs.oracle.com/javase/8/docs/api/java/lang/Short.html

2.2.3.2 O paquete java.util

java.util.Date

A clase `Date` representa un instante específico no tempo, con precisión de milisegundos. Ofrece métodos de comparación de datas, de obtención e modificación da data en formato

de milisegundos desde a orixe dos tempos UNIX (1 de xaneiro de 1970). Estes son os seus construtores e os métodos máis salientables.

Construtor	Descrición
Date()	Crea un obxecto Date e o inicializa coa data e hora actuais, con precisión de milisegundos.
Date(long date)	Crea un obxecto Date e o inicializa coa data e hora indicadas no parámetro date, representada coma o número de milisegundos transcorridos desde o 1 de xaneiro de 1970.
Método	**Descrición**
after()	Comproba se unha data é posterior á do obxecto.
before()	Comproba se unha data é anterior á do obxecto.
compareTo()	Compara dúas datas para ordenación.
equals()	Comproba se dúas datas son iguais.
getTime()	Devolve o número de milisegundos desde o 1 de xaneiro de 1970.
toString()	Convirte a data en cadea de texto.

Por exemplo:

```
Date data = new Date();
System.out.println(data.toString());
```

Pode obterse máis información da clase `Date` na documentación oficial de Oracle https://docs.oracle.com/javase/8/docs/api/java/util/Date.html

java.util.Random

Esta clase emprégase para xerar un fluxo de números pseudo-aleaorios. Un número pseudo-aleatorio é un número xerado nun proceso que parece producir números ao azar, pero non o fai realmente. As secuencias de números pseudo-aleatorios non amosan ningún patrón ou regularidade aparente desde un punto de vista estatístico, a pesar de ser xeradas por un algoritmo completamente determinista, no que as mesmas condicións iniciais producen sempre o mesmo resultado. Para inicializar dito algoritmo é preciso un valor inicial coñecido como semente.

Construtor	Descrición
Random()	Crea un novo xerador de números aleatorios, inicializando a semento a un valor que é altamente probable que sexa distinto ao de calquera outra invocación deste construtor.
Random(long seed)	Crea un novo xerador de números aleatorios, a partir da semente que se lle indica como parámetro.
Método	**Descrición**
nextInt()	Devolve o seguinte número enteiro pseudo-aleatorio.
nextLong()	Devolve o seguinte número enteiro longo pseudo-aleatorio.
nextBoolean()	Devolve o seguinte número booleano pseudo-aleatorio.
nextFloat()	Devolve o seguinte número real pseudo-aleatorio.
nextDouble()	Devolve o seguinte número real dobre pseudo-aleatorio.
nextGaussian()	Devolve o seguinte número pseudo-aleatorio nunha distribución normal Gaussiana.

Por exemplo:

```
Random numeroAleatorio = new Random();
for (int i=0; i<10; i++) {
    System.out.println(numeroAleatorio.nextInt());
}
```

Pode obterse máis información da clase `Random` na documentación oficial de Oracle https://docs.oracle.com/javase/8/docs/api/java/util/Random.html

Realiza a tarefa 3.2. "Creación dun programa que empregue a clase Random", onde se empregará a clase `Random` para simular un dado de 6 caras.

2.3 Tarefas

As tarefas propostas para esta actividade son as seguintes:

- Tarefa 1.1. **Creación dun programa que empregue a clase `Math`**, onde se empregará a clase `Math` para calcular as raíces dun polinomio de segundo grao.
- Tarefa 1.2. **Creación dun programa que empregue a clase `Random`**, onde se empregará a clase `Random` para simular un dado de 6 caras.

2.3.1 Tarefa 1.1. Creación dun programa que empregue a clase Math

Nesta tarefa empregarase a clase `Math` para calcular as raíces dun polinomio de segundo grao.

Enunciado

Crea un programa en Java no que se calculen as raíces reais dun polinomio de segundo grao a partir dos seus coeficientes a, b e c indicados en cadansúa variable.

Solución

```
package polinomio2;

public class Polinomio2 {
    public static void main(String[] args) {
        double a = 2;
        double b = 3;
        double c = -1;

        double dentroRaiz = Math.pow(b,2) - 4*a*c;
        if (dentroRaiz < 0) {
            System.out.println("Non hai solucións reais");
        } else {
            double solucion1 = (-b + Math.sqrt(dentroRaiz)) / (2*a);
            double solucion2 = (-b - Math.sqrt(dentroRaiz)) / (2*a);
            System.out.println("Las soluciones son " + solucion1 + " y " + solu-
cion2);
        }
    }
}
```

2.3.2 Tarefa 1.2. Creación dun programa que empregue a clase Random

Nesta tarefa empregarase a clase `Random` para simular un dado de 6 caras.

Enunciado

Crea un programa en Java que simule o lanzamento dun dado de 6 caras en 10 intentos. Que modificación habería que facer para que en todas as execucións o resultado fose sempre a mesma secuencia.

Solución

```
package dado;

import java.util.Random;

public class Dado {
    public static void main(String[] args) {
        int min = 1;
        int caras = 6;
        Random aleatorio = new Random();
        for(int i = 0; i < 10; i++) {
            int tirada = aleatorio.nextInt(caras)+min;
            System.out.println(tirada);
        }
    }
}
```

Neste caso, a semente inicialízase de forma automática, polo que toma valores diferentes en cada execución. Para obter sempre a mesma secuencia, o que resulta moi útil para a detección de erros nos programas, deberíamos ser nós o que iniciaramos a semente a un valor determinado, para o que unicamente habería que cambiar a chamada ao construtor da clase `Random` para incluír a semente:

```
Random aleatorio = new Random(1314);
```

3. Materiais

3.1 Documentos de apoio ou referencia

- Especificación da linguaxe Java: http://docs.oracle.com/javase/specs/
- Java SE 8 API Documentation: https://docs.oracle.com/javase/8/docs/api/index.html
- The Java Tutorials: https://docs.oracle.com/javase/tutorial/
- V. RESÚA EIRAS. POO Java. http://iespazodamerce.es/wiki/index.php?title=POOJava
- Java static. Atributos y métodos estáticos o de clase. http://puntocomnoesunlenguaje.blogspot.com.es/2013/02/java-static.html
- J. SÁNCHEZ. Manual de Java. http://jorgesanchez.net/java
- P.A. SZNAJDLEDER. Java a fondo. Alfaomega. 2ª edición.
- Ejercicios propuestos y resueltos programación orientado a objetos Java. https://www.discoduroderoer.es/ejercicios-propuestos-y-resueltos-programacion-orientado-a-objetos-java/
- J. BOBADILLA SANCHO. Java a través de ejemplos. Ra-Ma. Ed. 2003.
- Clases básicas predefinidas. Depto. Lenguajes y Ciencias de la Computación. E.T.S.I. Informática. Universidad de Málaga. http://www.lcc.uma.es/~vicente/docencia/poo/teoria/poo_4_cbasic.pdf
- Wikipedia. https://www.wikipedia.org

3.2 Recursos didácticos

- Ordenador persoal con conexión a Internet.
- Contorno de desenvolvemento NetBeans.
- Java SE Development Kit.
- Apuntamentos da profesora.
- Proxector.

4. Avaliación

Criterios de avaliación seleccionados para esta actividade (exemplo)	**Evidencia de aprendizaxe**	**Instrumento de avaliación** (tomados da aplicación de programación)	**Peso na cualificación da UD**
▪ CA2.3. Instanciáronse obxectos a partir de clases predefinidas.	▪ Proxecto Java	▪ OU 1. Táboa de indicadores sobre proxectos Java nos que se instancien obxectos a partir de clases predifinidas.	10%
▪ CA2.5. Escribíronse chamadas a métodos estáticos.	▪ Proxecto Java	▪ OU 2. Táboa de indicadores sobre un proxecto Java no que se escriban chamadas a métodos estáticos.	10%
▪ CA2.7. Incorporáronse e utilizáronse librarías de obxectos.	▪ Proxecto Java	▪ OU 3. Táboa de indicadores sobre proxectos Java nos que se incorporen e empreguen librarías de obxectos.	15%

4.1 Modelo de proba

4.1.1 CA2.5. Escribíronse chamadas a métodos estáticos.

Enunciado

O teorema do coseno é unha xeneralización do teorema de Pitágoras para triángulos non rectángulos que relaciona un lado dun triángulo cos outros dous e co coseno do ángulo formado por estes dous lados.

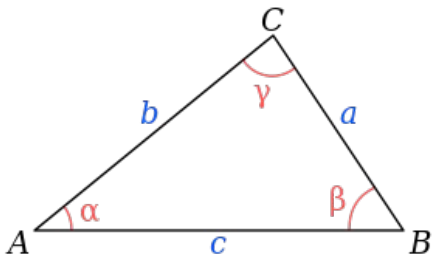


Figura 1: Notación habitual dun triángulo (fonte: Wikipedia)

Dado un triángulo ABC, sendo α, β, γ, os ángulos, e a, b, c, os lados respectivamente opostos a estes ángulos, entón:

$$c^2 = a^2 + b^2 - 2ab\cos(\gamma)$$

Polo que para calcular c:

$$c = \sqrt{a^2 + b^2 - 2ab\cos(\gamma)}$$

Crea un programa en Java que permita calcular o lado c a partir dos outros dous lados e o ángulo forman.

Solución

```
package trigonometria;
```

```
public class Trigonometria {

    public static void main(String[] args) {
        double a = 12;
        double b = 15;
        double gamma = Math.PI/4; //Medida do ángulo en radiáns
        double c;
        c = Math.sqrt(Math.pow(a,2) + Math.pow(b,2) - 2 * a * b * Math.cos(gam-
ma));
        System.out.println(c);
    }
}
```

Táboa de indicadores

OU 7. Táboa de indicadores sobre un proxecto Java no que se escriban chamadas a métodos estáticos.

Nome:		Data:
CA2.5. Escribíronse chamadas a métodos estáticos.		

Indicadores	Puntuación base	Puntuación obtida	Observacións
▪ Empregou correctamente a constante PI da clase Math.	2		
▪ Empregou correctamente o método pow da clase Math.	2		
▪ Empregou correctamente o método sqrt da clase Math.	2		
▪ Empregou correctamente o método cos da clase Math.	2		
▪ Fixo correctamente o cálculo da fórmula.	2		

4.1.2 CA2.3. Instanciáronse obxectos a partir de clases predefinidas. CA2.7. Incorporáronse e utilizáronse librarías de obxectos.

Enunciado

- Crea un programa en Java que calcule unha aposta aleatoria á lotaría primitiva. Deberá xerar 6 números entre 1 e 49 e un reintegro entre 0 e 9.
- Crea un programa en Java que cree un obxecto Date coa data actual e o amose por pantalla.

Solución

- Lotaría primitiva.

```
package primitiva;

import java.util.Random;

public class Primitiva {

    public static void main(String[] args) {
        int numeros = 6;
        int min = 1;
        int max = 49;
        int maxReintegro = 9;
        Random aleatorio = new Random();
        for(int i = 0; i < numeros; i++) {
```

```
            int numero = aleatorio.nextInt(max)+min;
            System.out.println(numero);
        }
        int reintegro = aleatorio.nextInt(maxReintegro);
        System.out.println("R: " + reintegro);
    }
}
```

- Data actual.

```
package data;

import java.util.Date;

public class Data {

    public static void main(String[] args) {
        Date data = new Date();
        System.out.println(data);
    }
}
```

OU 6. Táboa de indicadores sobre un proxecto Java no que se instancien obxectos a partir de clases predefinidas.

Nome:		Data:	
CA2.3. Instanciáronse obxectos a partir de clases predefinidas.			
Indicadores	**Puntuación base**	**Puntuación obtida**	**Observacións**
▪ Instanciouse un obxecto da clase Random.	5		
▪ Instanciouse un obxecto da clase Date.	5		

OU 8. Táboa de indicadores sobre un proxecto Java no que se incorporen e empreguen librarías de obxectos.

Nome:		Data:	
CA2.7. Incorporáronse e utilizáronse librarías de obxectos.			
Indicadores	**Puntuación base**	**Puntuación obtida**	**Observacións**
▪ Incorporouse correctamente a clase Random.	1		
▪ Incorporouse correctamente a clase Date.	1		
▪ Empregouse correctamente o construtor da clase Random.	1		
▪ Empregouse correctamente o construtor da clase Date.	1		
▪ Empregouse correctamente algún método da clase Random para a xeración de números aleatorios.	2		
▪ Resolveuse o problema da lotaría axeitadamente.	3		
▪ Amosouse a data adecuadamente.	1		