public class SofaManager {
private Sofas[] sofas;
public SofaManager(Sofas[] sofas){
this.sofas = sofas;
}

// Create a constructor that creates set from a given array of furniture

public FurnitureSet(Furniture [] someSet){
this.someSet = someSet;
}

// Add a furniture to array . If there is already the same piece of furniture do not add it but throws an exception with a message ("This piece of furniture is already in the set ")

public void addFurniture(Furniture furniture) throws MyException {
for(int i=0; i< someSet.lenght;i++){
if(someSet[i] != null){
if(someset[i].equals(furniture)){
throw new MyException("This piece...");
}
else{
someSet[i] = furniture;
}
}

// Override the toString don't use Array.toString() but super from the superclass

@Override
public String toString() {
super.toString() + "seats:" + seats;

}

//method that checks if a sofa from the set is if greate that given length and width, then change sofa type to "U-shape"

public void setSofaType(int length, int width) {
for(int i = 0; i< someSet.lenght; i++){
if(someSet[i] instanceof Sofa){
Sofa sofa = (sofa) someSet[i];
if(sofa.getLength() > lenght && sofa.getWidth() > width)
sofa.setTypeSofa(lenght, width);
}
}

Additional: create methods: *numberTables* that returns a number of tables in the set and method *reorganizeFurniture* that moves all the tables in the beginning and the sofas to the end of the set.

package UserPackage;

Do the necessary imports:

public class Main {
public static void main(String[] args) {

//Create a table
Table table = new Table(. . .);
//Create two sofas: sofa and sofa 2
Sofa sofa = new Sofa(. . .); ...
//Create a set of furniture
FurnitureSet furnit = new FurnitureSet();
//add to this set the table and sofa and sofa2 to this set. (check for a possible exception)
try{
someset.addFurniture(table);
...
}catch(Exception e){
sout(message e){

// create array of furniture furnArray that contains table and sofa -> Furniture[] funArray = {table, sofa};

//create a furniture set furnitureSet2 using the furnArray as a parameter for the constructor
↳ FurnitureSet furniture2 = new FurnitureSet(furnarray);
//print information about the furniture inside the set

// use setSofaType method for length 20 and width 30;
//create a raw arraylist furnitureList and add table, sofa and sofa2

//print information about the furniture using a loop for the arrayList

// *use methods numberTables and reorganizeFurniture*

Clone method

```
public *class* clone(){
 *class* clonedEcosystem = new EcoSystem(this.rows, this.columns);
  clonedEcosystem = new Animal[rows][columns]
    for(int i...)
      for(int j...)
        if(clonedEcosystem.animals[i][j] instanceof Bird){
          { if...  clonedEcosystem.animals[i][j] = new Bird(this.animals[i][j].getSpecies(), this.animals.getIsm
  }
  return clonedEcosystem
public EcoSystem clone(){
EcoSystem clonedEcoSystem = new EcoSystem(this.rows, this.columns);
clonedEcoSystem.animals = new Animal[rows][columns];
 for(...)
   for(...)
    if(this.animal[i][j] instance of Bird){
       clonedEcosystem.animals[i][j] = new Bird(this.animal[i][j].getspecies(),...)
 return clonedEcosystem
```

Encapsulation: Data hiding, access control
"Private" = not accessible from a static context

static: thing shared in every class or subclass

Array = null / Default values
int = 0

Class: Models real world behaviour

Inheritance: Defines hierarchical relations among the objects of some class

Public → Para todos, sin restricciones

Protected → Para los de su jerarquia/herencia

```
public class Ejemplo {
    protected int numero;
    protected void metodoProtegido() {
        // Metodo accesible desde esta clase, subclase y paquete
    }
}
public class Subclase extends Ejemplo {
    public void otrometodo() {
        numero = 5
    }
}
```

Private → SOLO para la misma clase

Método equals

```
public boolean equals(Object o) {
    if (this == o) return true;
    if (o == null || getClass() != o.getClass()) return false;
    User user = (User) o
    return username.equals(user.username) y email.equals(ese.email);
public boolean equals (Object o) {
    if (this == o) return true;
    if ( o == null || getClass() != o.getClass()) return false;
    User user = (User) o;
    return username.equals(user.username) && email.equals(user.email);
protected Object clone() throws CloneNotSupportedException {
    Dog dog = (Dog) super.clone();
    if (puppies != null) {
    dog.puppies = new Dog[puppies.lenght];
    for (int i = 0; i < puppies.lenght; i++) {
        dog.puppies[i] = (Dog) puppies.[i].clone();
    }
    return dog; }
```

→ Bellman optimality

```
protected object clone() throws cloneNotSupported Exception}
  Dog dog = (Dog) super.clone();
  if (puppies != null)}
    dog.puppies: new Dog[ppppies.lenght]

protected Object clone() throws clone Not Supported Exception}
  Dog dog = (Dog) super.clone();
  if (puppies != null)}
  Dog.puppes (Dog) [ppppies.lenght)
  for(int i = 0; i < ppies.lenght; i++)
   dog.ppies[i]:(dog) puppies.clone(i);
protected Object clone() throws clone Not Supported Exception}
  Dog dog = (Dog) super.clone()
  if (puppies != null);
   dog.puppes = new Dog[ppies.lenght]
  for (.....))
   dog.puppies[i] = Dog[puppies[i].clone
```

Implements ↝ interface

~~Extends~~ Exception ↝ type checked

RuntimeExpetion

try}

{ catch (message e)

```
if (users.equals (user))
 thrau new My Exception ("Error"
```

Data types

'A' + 1 = 'B' ⇒ 65 + 1 = 66

Integer. int, numbers without decimals
Floating-point. float, decimals
Character. char, letras

/ Division → división de a/b
% Reminder → resto de a/b

```
int x, y, result;
x = 5;
y = 2;
result = x / y;   → result = 2
result = x % y;   → result = 1
```

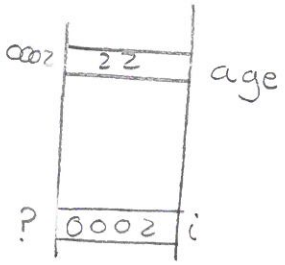
```
double result;
int a;
a = 45;
result = ((double) a / 12);
result = 22'5
```

```
x = 10;
y = ++x;
y = 11
```

```
x = 10;
y = x++;
y = 10
```

AND &&
OR ||
NOT !

Pointers!!!




```
int* i;
int age = 22;
i = &age;   i → 0002
```
↳ Apunta a la dirección de memoria donde se almacena el valor de "age", pero no al valor

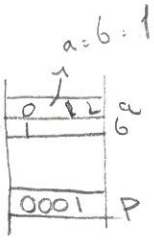
```
d *i ? = 23
```
↳ Accede a la dirección de memoria de i y toma el valor almacenado

```
*i = *i + 1;
```
↳ Suma 1 al valor almacenado en la dirección de memoria de i

Example

```
int count = 0
int *m;
m = &count;
```
} pone en m la dirección de memoria de count

```
#include <stdio.h>
Void main ()
{
  int a = 0, b = 1;
  int *p
  p = &a;
  printf("%x", &a);   → dir. memoria a
  printf("%x", p);    → dir memoria a (almacenada en p)
  printf("%d", a);    → a = 0
  a = b;
  printf("%d", *p);   →
  *p = 2;             → modifica el valor de donde apunta (a)
  printf("%d %d", a, b);   → 2,1
```



*p = toma el valor real a donde apunte

723

remainder: 723 % 10 = 3

723 = 723 : 10 = 72

Funciones C.

```
#include <stdio.h>
int main () {
    int x, y, result;
    printf("write the 'x' value ");
     scanf("%d",&x);
    printf("write the 'y' value ");
     scanf("%d",&y);
    result = multiply(x,y);
    printf("The result of multiplying %d and %d is %d",x,y,result);
    return 0;
int multiply(int x, int y){
    int result = 0;
    for(int i; i<=y; i++){
     result = result +y;
     }
    return result;
{
```

```
#include <stdio.h>
Void FuncExample(int* vf1, int vf2);        (int* a, int b);
 Int main(void){
 int a = 5;
 int b = 10;
 FuncExample(&a, b);                        (&b, b);
 print("(main) -> %d %d/n", a,b);  ⟶  7, 10 main
}
void FuncExample (int* vf1, int vf2) {
 *vf1 += 2;                                  10   10
 vf2 -= 2;
  printf("(function) -> %d %d\n", *vf1,vf2);
 {
```

7, 5 func

```
float squreOfTheDiff(float a, float b){
 float dif;
   dif= a-b;
   return dif* dif;


# include <stdio.h>
int isEven(int num){
    if(num %2 == 0){
      return 1;
     }else{
        return 0;
     }
}
void main(){
  int number;
  printf("Ente a num:");
  scanf("%d",& number);
  if (is even(member))
  else{
  }
```

C. Programming

Functions

```
#include <stdio.h>
Void functionName()}
    . . .
    . . .
{
int main()}
    . . .
functionName();
    . . .
{
```

with parameter

```
# include <stdio.h>
Void calculateSquare (int number)}
      int square = number * number;
      printf("Square of %d is %d \n", number, square);
{
int main ()}
  calculate Square(5);
  return 0;
{
```

Function prototype

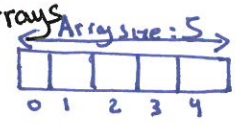
```
# include <stdio.h>
Int addNumbers(int number1, int number2);
Int main ()}
   int result = addnumbes(8,9);
  printf("The result is %d", result);
{
int addNumbes(int number 1, int number2)}
   int sum = number1 + number2,
   return sum;
   printf("After return "); #NO PRINT
{
```

User defined functions

Recursion

```
#include <stdio.h>
Int reursion (int n);
Void main()}
   printf("Introduce the num to sum");
   int n, result;
   Scanf("%d",&n);
   result = recursion(n);
  printf ("The sum is %d", result);
   {
Int recursion(int n)}
    if(n != 0)}
      return n + recursion(n-1);
    {else}
     return n;
    {
   {
```

Arrays



dataType arrayName [arraySize];

↳ float mark[5];

int mark[5] = {19, 10, 8, 17, 9};

↝ Escribir y vaciar al revés array

```
#include<stdio.h>
int main(){
  int nona[5];
 for(int i=0; i<=4;i++){
   printf("introduce the value %d of the array", i);
   scanf("%d",& nona[i]);
   {

for(i=4; i>=0;i--){
  printf(" The value in the position %d is %d\n", i, nona[i]);
```

↝ Hacer un average con array

```
#Include <stdio.h>
 void main(){
   int array[5]
   int sum=0;
   int average=0;
   int n=0;
 for(int i=0; i<=6; i++){
   printf(" Introduce the %od value", i);
   scanf(" %od",& array[i]);
   n++;
   sum = sum + array[i];
 {
 printf("The sum is: %od", sum);
 average = sum/n;
 printf("The average is: %od", average);
{

int c[10]={1,2,3,4,5,6,7,8,9,10};
void main(){
  int a, b=0;
  for (a=0; a<10; ++a)
   if((c[a]%2)==1) b+= c[a];
  printf ("%d", b);
```

1 3 5 7 9

Exercise random

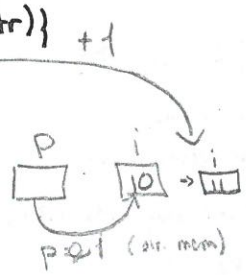
```
#include <stdio.h>
void display(int arra[], int size);
void display(int arra[], int size){
  int sum = 0;
  int minValue = arra[0];
  int maxValue = arra[0];

  printf("Sumatory \n");
  for (int i = 0; i < 6; i++){
    sum = sum + arra[i];
  {
printf("The sum is %d", sum);
printf("Min. value");
  for(int i = 0; i < 6; i++){
  if(minValue > arra[i]){
    minValue = arra[i];
printf("Min value: %d", minValue);
printf("Max value");
  for(int i = 0; i < 6; i++){
    if(maxValue < arra[i]){
      maxValue = arra[i];
    {
  {
printf("Max. Value: %d", maxValue);
{
```

```
int main(){
int arra[] = {3, 4, 5, 6, 7, 8};
int size = 6;
display(arra, size);
return 0;
{
```

Functions and pointers

```
#include <stdio.h>
void swap(int *n1, int *n2);
int main()
 int num1=5, num2=10;
swap(&num1,&num2)
printf("num1 = %d\n", num1);
printf("num2 = %d ", num2);
return 0;
{
void swap(int* n1, int* n2);
 int temp;
 temp = *n1;
*n1 = *n2;
*n2 = temp;
```

```
void addone(int* ptr)}
 (*ptr)++;
{
int main()}
int *p, i = 10;
p = &i;
addone(p);
printf("%d",*p);=11
return 0;
```



Strings

```
char c[] = "c string";
```


```
char s[5]:
```

STRINGS acaban en '\0'
para espacios, char a = ' ';

```
#include <stdio.h>
void displayingString(char str[]);
int main()}
char str[50];
printf("Enter string");
fgets(str, sizeof(str), stdin);
displayingString(str);
 return 0;
void displayingString(char str[])}
 puts(str);
 {
```

strcpy()
strlen()
strcomp()

Length of an string

```
int main()}
 char str[] = "Programming is fun";
 for(int i = 0; str[i] != '\0'; i++);
  printf("count of %d ", i);
  return 0;
   {
```

✧ Structs

```
Typedef Struct{
 int surtNum;
 char matricula[];
 Carburante myCarburante;
 float repost;

}Carburante;

Typedef Struct{

}Ventas;
```

✧ Multidimensional Arrays

float x[3][4];

	Column 1	Column 2	Column 3	Column 4
Row 1	x[0][0]	x[0][1]	x[0][2]	x[0][3]
Row 2	x[1][0]	x[1][1]	x[1][2]	x[1][3]
Row 3	x[2][0]	x[2][1]	x[2][2]	x[2][3]

Way to initialize two-dim. array

int c[2][3] = {{1,3,0}, {-1,5,9}};

```
#include <stdio.h>
int city =2;
int week 7;
int temperature[city ][week];
for(int i=1; i<=city ;i++){
  for(int j=1;j<=week;j++){
  printf(" city: %d week: %j => ");
  scanf("%d", tempeature[i][j]);
{
for(int i=1; i<=city, i++){
  for(int j=1;j<=week;j++){
  printf("City: %od, week: %od -> %od", i, j, tempeature[i][j]);
  }
{
return 0;
{
```

✧ Array + func

```
#include <stdio.h>

int fact(num[2][2]);   (→ int)
void main(){
 int num[2][2];
 for(int i=0; i<=1; i++){
   for(int j=0;j<=1;j++){
   printf("Introduce the valle of a %od %od", i, j);
   scanf("%d", &num[i],[j]);
  }
{
 fact(num);
{
fact (int num[][2]){
  for(int i=0; i<=1; i++){
    for(int j=0; j<=1;j++){
    printf("%d\t, num[i][j]);
     if(j==1){
    printf("\n");
    {
{ {
```

Pointers

```
int var=5;
printf("%d", var); = 5
printf("%d", &var); = -33604...
```

Syntax

```
int* p;
```

```
int *pc, c;
c=5;
pc=&c;
printf("%d", *pc);
```

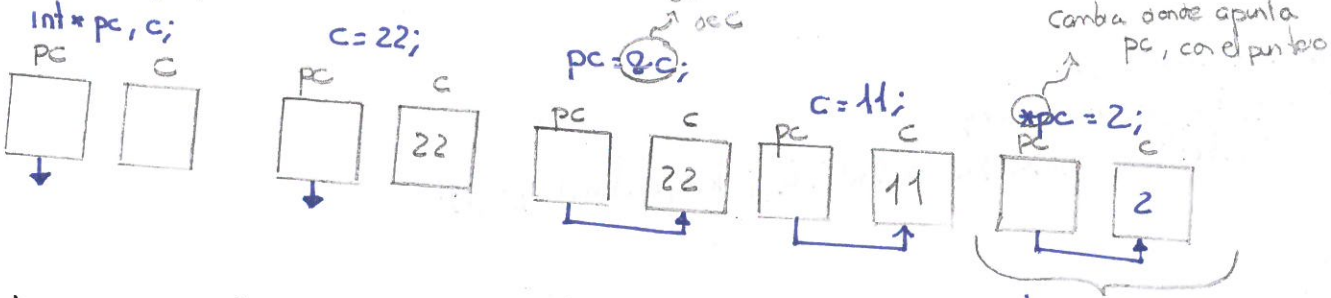
accede al valor de la dir. de memoria

```
int main(){
 int* pc, c;
 c=22;
 printf("Address of c: %p\n", &c); = 22643...
 printf("Value of c: %d\n\n", c); = 22
```

Imprimir dir memoria

```
int* pc, c;
c=5;
pc=&c;
c=1;
printf("%d", c); = 1
printf("%d", *pc); = 1
```

dir. memoria




```
int c=5;
int *p=&c;
printf("%d", *p); = 5
```

Arrays and pointers

```
Int main(){
  Int i, x[6], sum=0;
  printf("Enter 6 numbers: ");
   for(i=6; i<6; i++){
    Scanf("%d", x+i);{ because dir. mem of x[0] = dir mem of x
   //Equal to x[i]
    Sum += *(x+1); //Equal to sum += x[i]
   {
```

&

```
Void main(){
  int x[5]={1,2,3,4,5};
  int* ptr;
  ptr=&x[2]; //A la dir. memoria del 3er valor de x[]
  printf("*ptr = %d\n", *ptr); = 3
  printf("*(ptr+1) = %d\n", *(ptr+1)); = 4
  printf("*(ptr-1) = %d\n", *(ptr-1)); = 2
  {
```

1)
```
#include <stdio.h>
void FuncionPrueba(int *a, int *b);
int main(void){
 int a = 5;
 int b = 10;
FuncionPrueba(&b,&a);   Pasa 10 y 5
 printf("(main) -> %d %d\n", a, b);
{
void FuncionPrueba(int *a, int *b){
*a = *a + 5;  = 15
*b = *b - 2  = 3
printf("(funcion) -> %d %d\n", *a, *b);
```

(funcion) -> 15 3

(main) -> 3 15

2)
```
#include <stdio.h>
int main(void){
 int i=0, x=0;
 i=2;
 for(i=2; i<=9; i++)}
 ++i;
 if(i % 4 == 0) x++;
  printf("%d " x);
 {
 printf("\nx = %d" x);
 {
```

```
#include <stdio.h>
int main(void){
 int i=0; x=0;
  i=2;
 while(i<10)}
 ++i
 if(i % 4 == 0) x++;
 printf("%d;" x);
 i++;
 {
 printf("\nx = %d ", x);
 {
```

3) Leer en C secuencia de caracteres, cuenta mayúsculas y minúsculas.

```
#include <stdio.h>
int main(){
int a=0, e=0, i=0, o=0, u=0;
char c;
printf("introduce each letter:\n");
while(c != '.'){
 switch(c){
  case 'a':
    a++;
   break;
  case 'A':
    a++;
   break;
  case 'e':
   e++;
   break;
    .
    .
    .
 {
{
printf("Vocales: a=%d. e=%d, i=%d, o=%d, u=%d.", a, e, i, o, u);
{return 0;
```

Control Statement. Unit 4

```
int number;
Scanf("%d",&number);
if(number==0){
  printf("The number is 0");
 }else if(number %2 ==0){
    printf("%d: even",number);
  }else{
    printf("%d: odd", number);
```

1/ Salida?

```
#include <stdio.h>
void Funcion(int*a, int*b);
int main(void){
 int x=2;
 int y=8;
 Funcion(&y,&x);
printf("(main)-> %od %od\n", x,y);
{
void Funcion(int *a, int*b){
   *a = *a*2;
   *b = *b/2;
  printf("(funcion)-> %od %od\n", *a,*b);
```

pasa dir. memoria

accede en dir. memoria

(función) -> 16 1
(main) -> 1 16

Automórfico

```
#include <stdio.h>
int main(){
  int n;
  printf("introduce el numero: \n")
   scanf("%od",&n);
   ¡Contamos digitos!
  int digitos=0;
  while(n!=0){
     digitos++;
    n= n/10;
   {
   Vemos si los ult. nos digitos son iguales
    int i=0;
    int iguales=1;
    for(i.0; i<=digitos; i++){

  {
```

4/ Función que lea numeros posit, negativos, hasta poner 0

```
#include <stdio.h>
Void Contador();
int main()
int positivos=0;
int negativos=0;
Contador(&positivos,&negativos);
printf("Hay %od positivos y %od negativos", positivos,negativos);
{
```

pasa la dirección de memoria

```
Void Contador (int *positivos, int *negativos){
int secuencia=1;
printf("Introduce numero a numero: \n");
while(secuencia!=0){
 scanf("%od",&secuencia);
 if(secuencia>0){
  *positivos = *positivos+1;
  {
 if(secuencia<0){
 *negativos = *negativos+1;
 { {{
```

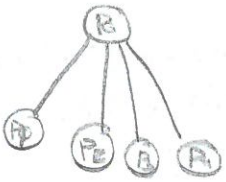
accede al valor de la dir. de memoria

#include

Exercise 1.

gcc Name.C -o execName

```
if(geteuid == 0){        Effective user. Root in this case
  int pid = getpid();
  for(int i=1; i<=8; i++){
    if(getpid() == pid){
      fork();
      wait(NULL);
  }
printf("PID: %d, PPID %d, EUID %d ", getpid(), getppid(), geteuid();
}else{                   User normal
  for(int i=1; i<=8; i++){
    if(fork()!=0){
      wait(NULL);
      break;
    }else{
      printf("PID: %d, PPID %d, EUID %d" getpid(), getppid(), geteuid();
    }
  }
```



Con fork()!=0 comprueba si es el proceso padre, y en caso de serlo lo "mata" y pasa al siguiente

Exercise 2.

```
int p=atoi(argv[1])
int array[6]; To keeep process IDs
array[0]=getpid(); For P0
printf("Exec starts: P0 PID %dPPID %d "getpid(); getppid());
for(int i=1; i<6; i++){
  if(fork==0){ Checks if child
    array[i]=getpid();
    printf("Exec start: PID PPID");
    sleep(i);
    printf(Exec end...")
    exit(0);
  }
}
printf("P0 waits for P%d",i);
waitpid(array[P], NULL, 0);
printf("Exec end: PID...");
```

1. Print start de todos
2. Proceso que esperará
3. Espera al último (3) y printea P0

gcc Name.c -lpthread execName

Exercise 3.

```
:
const char *words[] = {"hello", "hello", "hello", "hello"};   Defining
double delays[] = {1, 0.5, 0.25, 0.125};
int num_words = 4;
:

for (int i = 0; i < num_words; i++) {
  data_word = words[i]
  data.delay = delays[i]                                      to store
if (pthread_create(&threads, NULL, print_word, (void *)&data) != 0) {
  perror("Failed to create thread");
   return 1;
  }
pthread_join(thread, NULL);                                   Get into one thread
:
void print_word(void *arg)
  {
   hilo *data = (hilo *) arg;
for(int i=0; data[i]; = '\0'; i++) {
   printf("%c", data->word[i]);                               Imprime la palabra y espera los segundos del delay
   fflush(stdout);
   sleep(data->delay);
   }
```

Exercise 4.

```
void *generateRandomNum(void *randomNumber) {
srand((unsigned int) time(NULL));                             genereates a random num (VERY HIGH!!)
int* param = (int*) randomNumber
int randValueInThread = rand() %15
*param = randValueInthread;
sleep(1);
return NULL;
{
int main {
thcanter = 0;
  comprobar arg = 2;
{
int numberToBeGuessed = atoi(argv[1]));
if(numTBG > 14 or  numTBG < 4) {
  erro;
  {
while(true) {
creates a process + joins calling random num;
  if (randomNumber == numberToBeGuessed) {
  threadcanter++;
  printf("It took %d times to guess it", theeadcanter); return 0;
{ else}
  threadCanter++;
{
{
```

UID = UserID
GID = GrapID
i_node = num de archivos y carpetas almacenados en las cuenta

```
}}

public void addService(PromotionalService service) {
for(int i =0; i < PromotionalService[] lenght; i++)}
 if(PromotionalService[i] == null)}
    PromotionalService[i] = service;
  {   services.

dd a user to the array if there is null . If there is already the same user (use equal
    function) do not add it but throw an exception MyException with the message ("User
    already exists.")
                                          ... throws MyException{
public void addUser(User user) {        for(int i=0; i < users.lenght; i++)}
                                         if(users[i] == null)}
                                            users[i] = user;
      }                                 {if(users[i].equals(user))}
public void runAllCampaigns() {           throw new MyException("Error, it already exists");
for(int i=0; i < services[i]; i++)}
   servicesEi]. runCampaign();
{
}

public double calculateTotalCost() {
  double totalCost=0;
for(int i=0; i < services.lenght; i++)}
 totalCost= totalCost + services.getCost();
 {
}return totalCost;

//calculate total revenue of the campaign
public double calculateTotalRevenue() {
  double totalRevenue=0;
for(int i=0; i < services.lenght; i++)}
totalRevenue= totalRevenue + services[i].getCost();
 {
 return totalRevenue;
}
@Override
public String toString() {
  return "PromotionalCampaign{" + "services=" + Arrays.toString(services) +", users=" + Arrays.toString(users) +'}'; }}
```

package UserPackage; [15 points]
public class MyException extends Exception {
  public MyException(String s) {
    super(s); }
}

→ De tipo "checked". Unchecked sería con "RuntimeException"
Se pueden definir igual

Do the necessary imports:

```
import UserPackage.User;
import PromotionalCompanyPackage.*;
```

```
import problemsExamIServ.PromotionalCompanyPackage.PromotionalCampaign;
import java.util.ArrayList;
import problemsExamIServ.PromotionalCompanyPackage.PromotionalService;

public class Main {
   public static void main(String[] args) {
PromotionalService emailMarketing = new EmailMarketing("EmailCampaign", 132);
PromotionalSeruce socialMediaMarketing= new socialMediaMarketing("SMC", 211);


,campaign.runAllCampaigns;
Sout("TotalCost" + campaign.CalculateTotalCost);
```

}

```
@Override
public void runCampaign() {  System.out.println("Running email marketing campaign..."); }
```

Complete the method `calculateRevenue()` such as it calculates a 100% increase of revenue

```
@Override
public double calculateRevenue() {
    return this.cost * 2;
}}
```

```
package UserPackage;
public class User {
    private String username;
    private String email;
    private static counter = 0;
    public User() {
        this.username = "User "+counter;
        this.email = username+"@example.com";
        counter++;
    }
    public User(String username, String email) {
        this.userName = userName;
        this.email = email;

    }

    public String getUsername() {      return username;   }
    public String getEmail() {       return email;   }
```

Usamos static porque queremos que el contador también sirva para las clases hijas y no solo un contador genérico

//define the type of the counter such as it is shared by all the instances of the class and calculates the current number of all users

Create the equals method: two user are equal when have the same names and emails

```
@Override
public boolean equals(Object o) {
    if (this == o) return true;
    if (o == null || getClass() != o.getClass()) return false;
    User user = (User) o;
    return username.equals(user.username) && email.equals(user.email);
```

Comprobamos que no viene vacío y que la clase del método coincide con la del objeto

Casting

Realizamos el método equals a cada elemento del objeto

```
ackage PromotionalCompanyPackage;
import  UserPackage.User;
import java.util.Arrays;

public class PromotionalCampaign {
    private PromotionalService [] services;
    private User [] users;

    public PromotionalCampaign() {
        this.users = new User[10];
    }
    public PromotionalCampaign(int n, PromotionalService [] services) {
```

Internet Marketing Campaign

There is a Java program for a promotional company that specializes in running marketing campaigns for internet shops. The company offers various services such as social media advertising, email marketing. The **CampaignRunnable** interface provides a contract for classes that can run marketing campaigns.The **PromotionalService** abstract class implements the *CampaignRunnable* interface and provides common functionality for promotional services, including a constructor and the *calculateRevenue*() method.
Subclasses **SocialMediaAdvertising, EmailMarketing,** inherit from **PromotionalService** and implement specific campaign running and revenue calculation logic.The class **PromotionalCampaign** creates and runs campaigns for user groups and includes different services.The **User** class represents users of the promotional company's platform. The *MyException* is a user-defined exception in the corresponding class.
Your task is fill in the gaps and add the necessary code samples.

```
package PromotionalCompanyPackage;

public interface CampaignRunnable {    void runCampaign();   }
```

```
Package PromotionalCompanyPackage;
public abstract class PromotionalService implements CampaignRunnable {
   protected String serviceName;
   protected double cost;

   public PromotionalService(String serviceName) {
      this.serviceName = serviceName;
   }
```

Implement a constructor that calls for the previous one:

```
   public PromotionalService(String serviceName, double cost) {
      super(serviceName);
      this.cost= cost;



   public String getServiceName() {   return serviceName; }
   public double getCost() { return cost; }
   public abstract double calculateRevenue();
}
```

```
Package PromotionalCompanyPackage;
public class SocialMediaAdvertising (String serviceName, double cost) {
```

Implement a constructor that calls for the constructor in the parent class:

```
   public SocialMediaAdvertising(String serviceName, double cost) {
      super(serviceName, cost);
```

↳ para la clase padre, super();

```
   }

   @Override
   public void runCampaign() {  System.out.println("Running social media advertising campaign..."); }
```

Complete the method `calculateRevenue()` *such as it calculates a 50% increase of revenue*

```
   public double calculateRevenue() {
      return this.cost* 1.5;
```

En este caso this.cost y "cost" tiene el mismo resultado, pero es buena práctica acceder a la instancia de memoria

```
   }}
```

```
package PromotionalCompanyPackage;
```

Implement a constructor that calls for the constructor in the parent class:

```
public EmailMarketing(String serviceName, double cost) {
   super(serviceName, cost);
```

```
// Override the toString method, don't use Array.toString()
public String toString() {
  return "Furniture } Type:"+type+" price:"+price "dimensions:"+width+weight+depth+"material:"+
material+ "discant:"+discant+"{";
}
//Override the equals method that compares two pieces of furniture
public boolean equals(Object o) {
   if(this == o) return false;
   if( o== null || getClass() != o.getClass()) return true;
   Furniture furniture = (Furniture) o;
} return type.equals(furniture.type)&& price.equals(furniture.price&& dimensions.equals(furniture.dimensions)&&.
```

Fill in the gaps for the Sofa class

```
package FurniturePackage;

public class Sofa extends Furniture {
  protected int seats;
  protected String type;
  private static sofaID = 0;

// Implement a constructor that calls for the constructor in the parent class:
  public Sofa(String type, double price, Dimensions dimensions, int seats) {
        super(type, price, dimensions);
        this.seats = seats;
}
```

//define the type of the *sofaID* such as it is shared by all the instances of the class and increments with each instance that is created (finally, contains a total number of instances)

Fill in the gaps for the Table class [15 points]

```
package FurniturePackage;
  public class Table extends Furniture {
  public int legs;
  Private static tableID= 0;

// Implement a constructor that calls for the constructor in the parent class:
   _____ Table(String type, double price, Dimensions dimensions, int legs){
    super(type,price, dimensions);
    this.legs = legs;
}
//Override the toString, don't use Array.toString() but super from the superclass
_____ String toString() {
   return super.toString()+"legs": +legs;


}
//Override the equals method
public boolean equals(Object o) {
    if(this== o) return true;
    if(o== null|| get.class() != o.getclass()) return false
    Fraible table = (Table) o;
} return type.equals(table.type)&& price.equals(table.price)&& legs.equals(table.legs);
```

//define the type of the table*ID* such as it is shared by all the instances of the class and increments with each instance created (finally:contains a total number of all instances)

Fill in the gaps for the FurnitureSet class

```
public class FurnitureSet {
   _____ Furniture[] set;
  this.set = new Furniture[20];
```

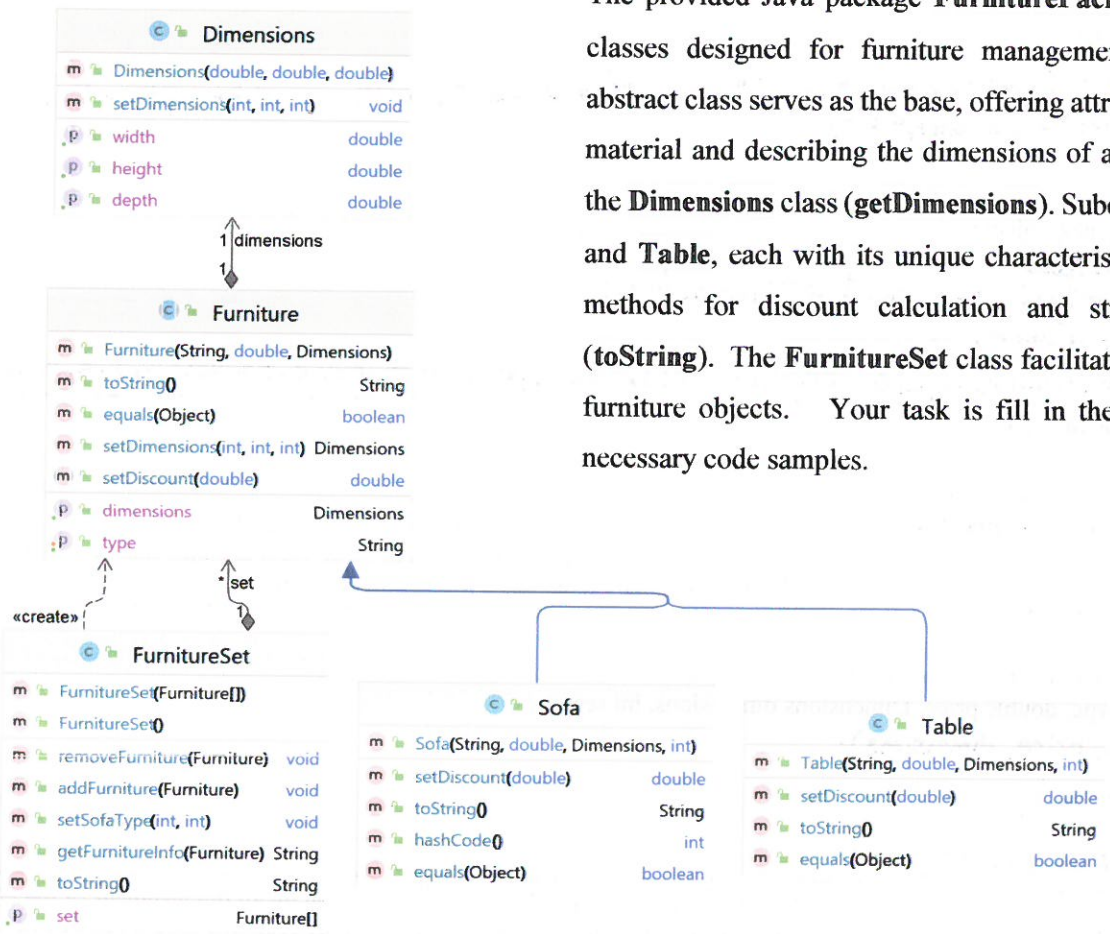
//change a default constructor such that it creates a set of 20 furniture items, randomly tables and sofas.

```
public FurnitureSet() {
  protected counter = 0;
  double random = Math.random();
```

```
ifor(int i=0; i< set.lenght; i++)}
  if(random <= 0'5)}
   Furniture[i] =
    set[i] = new Sofa("...", 10,...);
```

```
}
{ if(random > 0'5)}
  set[i] = new Table("...", 10,..
{ {
```

Furniture



The provided Java package **FurniturePackage that** contains classes designed for furniture management. The **Furniture** abstract class serves as the base, offering attributes like type, and material and describing the dimensions of a furniture piece via the **Dimensions** class (**getDimensions**). Subclasses include **Sofa** and **Table**, each with its unique characteristics and overridden methods for discount calculation and string representation (**toString**). The **FurnitureSet** class facilitates managing sets of furniture objects. Your task is fill in the gaps and add the necessary code samples.

```
public class Dimensions {
   protected double width, height, depth;
   public Dimensions(double width, double height, double depth) {
      this.width = width;       this.height = height;       this.depth = depth;     }
   public double getWidth() {       return width;    }
   public double getHeight() {       return height;    }
   public double getDepth() {       return depth;    }
   public void setDimensions(int width, int height, int depth) {
      this.width = width;       this.height = height;       this.depth = depth;     }
}
```

Fill in the gaps for the **Furniture class**

```
package FurniturePackage;
//think about the visibility modifiers you would give to the attributes
public abstract class Furniture{
  protected String type;
  protected double price;
  protected Dimensions dimensions;
  protected String material;
  protected double discount;
// Implement a constructor
   public Furniture(String type, double price, Dimensions dimensions) {
      this.type = type;
      this.price = price;
}     this.dimensions = dimensions;
   public String getType() {       return type;    }
   public double getPrice() {       return price;    }
   public Dimensions getDimensions() {       return dimensions;    }
   public Dimensions setDimensions(int width, int height, int depth) {       this.dimensions.setDimensions(width, height, depth);
     return this.dimensions;    }
   public abstract double setDiscount(double discount);
```

Sólo accesible desde la clase y sus hijas

Public pq necesita ser accesible desde cualquier lugar y tener los datos