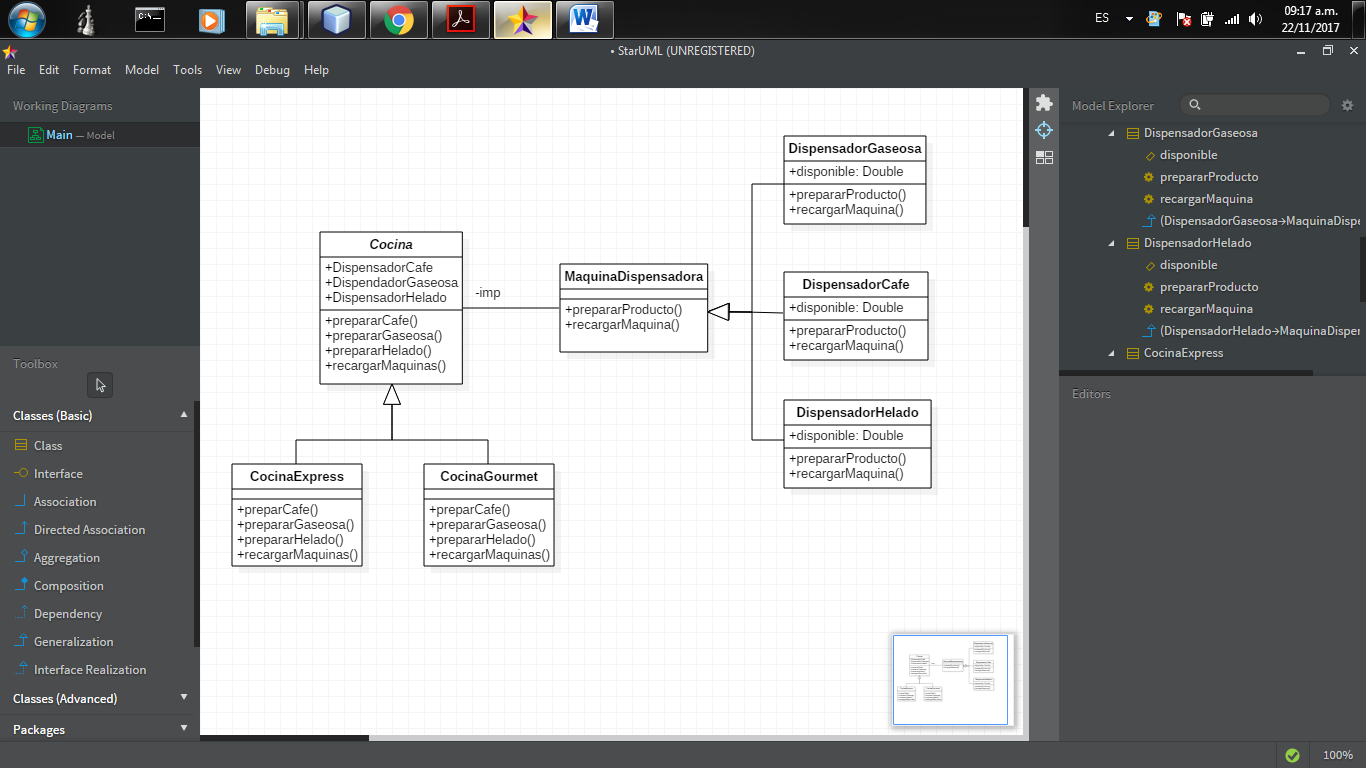
**ESCENARIO:**

Planteamos un escenario que sucedería dentro de un restaurante, la situación problema se presenta para el control de las máquinas dispensadoras de acompañamientos, en diferentes tipos de cocina, evitando un enlace permanente entre la cocina y la máquina que deseamos activar.



public class BridgeDispensadores {

public static void main(String[] args) {

Cocina cocina = null;

String tipoProducto;

int opcion, salir = 0, cantidad;

System.out.println("-------------------------");

System.out.println("BIENVENIDO AL RESTAURANTE");

System.out.println("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

try {

opcion = Integer.parseInt(JOptionPane.showInputDialog(null, "Tipo De Cocina:" + "\n"

+ "1.Cocina Express" + "\n"

+ "2.Cocina Gourmet"));

switch (opcion) {

case 1:

cocina = new CocinaExpress();

break;

case 2:

cocina = new CocinaGourmet();

break;

}

} catch (NumberFormatException ex) {

System.out.println("Parametros No Validos!"); }

while (salir != 1) {

System.out.println("-------------------------");

opcion = Integer.parseInt(JOptionPane.showInputDialog(null, "¿Què accion desea realizar?" + "\n"

+ "1.Preparar Cafe" + "\n"

+ "2.Preparar Gaseosa" + "\n"

+ "3.Preparar Helado" + "\n"

+ "4.Recargar Maquinas" + "\n"

+ "5.Salir" + "\n"));

switch (opcion) {

case 1:

tipoProducto = JOptionPane.showInputDialog(null, "Tipo de Cafe: ");

cocina.prepararCafe(tipoProducto);

System.out.println("Cafe: " + tipoProducto + "...Listo");

break;

case 2:

tipoProducto = JOptionPane.showInputDialog(null, "Tipo de Gaseosa: ");

cocina.prepararGaseosa(tipoProducto);

System.out.println("Gaseosa: " + tipoProducto + "...Listo");

break;

case 3:

tipoProducto = JOptionPane.showInputDialog(null, "Tipo de Helado: ");

cocina.prepararHelado(tipoProducto);

System.out.println("Helado: " + tipoProducto + "...Listo");

break;

case 4:

System.out.println("RECARGANDO MAQUINAS DISPENSADORAS...");

cantidad = Integer.parseInt(JOptionPane.showInputDialog(null, "Cantidad de LITROS a recargar: "));

cocina.recargarMaquinas(cantidad);

break;

case 5:

salir = 1;

break;

}

}

}

}

public abstract class Cocina {

protected MaquinaDispensadora maquinaCafe;

protected MaquinaDispensadora maquinaGaseosa;

protected MaquinaDispensadora maquinaHelado;

public Cocina() {

maquinaCafe = new DispensadorCafe();

maquinaGaseosa = new DispensadorGaseosa();

maquinaHelado = new DispensadorHelado();

}

public void prepararCafe(String tipo) {

maquinaCafe.prepararProducto(tipo);

}

public void prepararGaseosa(String tipo) {

maquinaGaseosa.prepararProducto(tipo);

}

public void prepararHelado(String sabor) {

maquinaHelado.prepararProducto(sabor);

}

public abstract void recargarMaquinas(double cantidad);

}

public class CocinaExpress extends Cocina{

public CocinaExpress() {

super();

}

@Override

public void recargarMaquinas(double cantidad) {

this.maquinaCafe.recargarMaquina(cantidad);

this.maquinaGaseosa.recargarMaquina(cantidad);

this.maquinaHelado.recargarMaquina(cantidad);

}

public class CocinaGourmet extends Cocina{

public CocinaGourmet() {

super();

}

@Override

public void recargarMaquinas(double cantidad) {

this.maquinaCafe.recargarMaquina(cantidad);

this.maquinaGaseosa.recargarMaquina(cantidad);

this.maquinaHelado.recargarMaquina(cantidad);

}

}

public interface MaquinaDispensadora {

void recargarMaquina(double cantidad);

String prepararProducto(String tipo);

}

public class DispensadorCafe implements MaquinaDispensadora {

public double disponible;

@Override

public void recargarMaquina(double cantidad) {

setDisponible(disponible + cantidad);

}

@Override

public String prepararProducto(String tipo) {

return tipo;

}

public double getDisponible() {

return disponible;

}

public void setDisponible(double disponible) {

this.disponible = disponible;

}

}

public class DispensadorGaseosa implements MaquinaDispensadora {

public double disponible;

@Override

public void recargarMaquina(double cantidad) {

setDisponible(disponible + cantidad);

}

@Override

public String prepararProducto(String tipo) {

return tipo;

}

public double getDisponible() {

return disponible;

}

public void setDisponible(double disponible) {

this.disponible = disponible;

}

}

public class DispensadorHelado implements MaquinaDispensadora {

public double disponible;

@Override

public void recargarMaquina(double cantidad) {

setDisponible(disponible + cantidad);

}

@Override

public String prepararProducto(String tipo) {

return tipo;

}

public double getDisponible() {

return disponible;

}

public void setDisponible(double disponible) {

this.disponible = disponible;

}

}

RESULTADO:

