a) ¿Cuántas clases se están definiendo en este ejercicio?

Se definen 3 clases: *Apostador*, *ComisionJuegoEspectaculos* y *Loteria*

b) ¿Para qué sirve la línea de código?

```
if __name__ == "__main__":
```

Lo que hace es revisar si el cdigo se está ejecutando directamente desde el archivo o si está siendo llamado por otro archivo. En caso de ser ejecutado directamente, se cumple el condicional.

c) ¿Qué sucede si retiro la línea de la pregunta anterior en nuestro código?, ¿Este sigue corriendo o hay error? Explique en ambos casos.

Si se elimina la condición y se remueve la indentación, el código correrá de la misma forma, ya que se sigue ejecutando desde el archivo directamente. Otro caso sería si se importase el archivo *main.py* sin el condicional en otro archivo, eso provocaria que se ejecutase el código de *main.py* por completo en la línea de importación.

d) ¿Cuántos objetos de la clase Apostador se están creando?

Se crean dos objetos de la clase *Apostador*.

e) ¿Cuáles objetos de la clase Apostador se están creando?

Se crean los objetos *apostador1* y *apostador2*.

f) ¿A quién está haciendo referencia la variable self de la línea 15 de la clase Apostador cuando se ejecuta el programa principal?

En esa línea se está pasando como argumento el objeto de clase *Apostador* con el que se invoque el método *play*.

g) ¿Cuántos objetos de la clase Loteria se están creando?

Se crean dos objetos de la clase *Loteria*, uno por cada vez que se invoca el método *play*.

- En la línea 4 del main.py cambiar el apostador1.deposit(500) por apostador1.deposit(300)
    - El programa avisa que se debe colocar más dinero en la wallet, ya que en el método play se coloca un argumento mayor al atributo *wallet*.

h) ¿Qué imprimiría el código por parte del apostador1?

Imprime: 300, apostador2.deposit(400), 300.

- En la línea 10 del main.py cambiar el apostador2.deposit(500) por apostador2.deposit(400)
    - En caso de que pierda la apuesta, queda con la cartera en cero.

i) ¿Qué imprimiría el código por parte del apostador2?

Imprime: 400, "Has ganado 720.0" o "Has perdido lo que apostaste", 720.0 o 0.

j) ¿Cuáles atributos de la clase Lotería están haciendo referencia a objetos?

El atributo *apostador* hace referencia a un objeto de la clase *Apostador*.

k) ¿Cuáles atributos de la clase Lotería están haciendo referencia a tipos primitivos?

El atributo *value* se asigna en *main.py* como un entero, pero también podría ser un flotante, ambos de tipo primitivo.

l) ¿Complete las siguientes líneas para que en la clase Loteria, se implemente el método de clase changeProbability?

```
_______
def changeProbability(__, nprobability):
    _______.probability = nprobability
```

Solución:

```
@classmethod
def changeProbability(clase, nprobability):
    clase.probability = nprobability
```

m) ¿Cómo sería la línea de código para llamar el método changeProbability?

```
loteria.changeProbability(0.75)
```

n) ¿Es correcto si en el método changeProbability que se creó, cambiar lo siguiente?

```
#Línea Original
- cls.probability = nprobability
#Línea Nueva
- Loteria.probability = nprobability
```

Si bien funcionará de la misma manera, se está colocando directamente la clase *Loteria*, por lo que si se hereda el método, solo cambiará los atributos de la clase *Loteria*.

o) ¿Cuántos métodos tiene la clase Loteria después de agregarle el nuevo método?

La clase *Loteria* queda con 5 métodos.

p) ¿Si el apostador1 gana el apostador2 también? Explique por qué pasa en caso de ser sí o no

No, los dos apostadores son objetos independientes, por lo que no se relacionan, no existe ningun método que llame a otro objeto de la clase *Apostador*, ni un atributo compartido más allá de la probabilidad inicial definida en la clase *Loteria*.

q) ¿Qué sucede si decido cambiar el atributo de clase probability a una constante? ¿Se considera correcto el uso del método changeProbability teniendo en cuanta este nuevo cambio?

No, la idea de una constante es tener un valor inmutable, tener un método que pueda modificar la constante sería contradictorio.

r) ¿Cuál es el tipo de retorno de los métodos gain() y commission() de la clase ComisionJuegoEspectaculos?

gain() es un método estático que retorna el valor numérico que gana la Comisión de juegos y espectáculos por la apuesta que hace el objeto de la clase *Apostador* por medio de la clase *Loteria*. *comission()*, al no ser un método estático, tiene acceso a los atributos de la clase y puede acceder a *COMMIPERCENTAJE*, valor numérico usado para calcular la comisión, y a *loteria*, el objeto que contiene el valor apostado.

s) ¿A quién está haciendo referencia la variable self de la línea 18 de la clase Loteria cuando se ejecuta el programa principal? ¿Podría omitirse el uso de la variable self en este caso?

No puede omitirse la variable *self*, ya que esta es necesaria para crear un objeto de la clase *ComisionJuegoEspectaculos*. Para poder inicializarse, esta clase necesita un objeto de la clase *Loteria*, y el *self* sirve para usar el objeto que ejecuta el método *playGame* como argumento.

t) ¿En la línea 15 de la clase apostador vemos como la clase recibe dos parámetros (value, self) especificar cuál de estos pasa por valor y cuál por referencia y por qué?

En este caso *value* pasa como valor, siendo una copia de un valor que debe ser nuérico para que el código funcione. Por otro lado, *self* es una referencia a un objeto de la clase *Apostador* que es necesario pasar como argumento para inicializar un objeto de la clase *Loteria*.