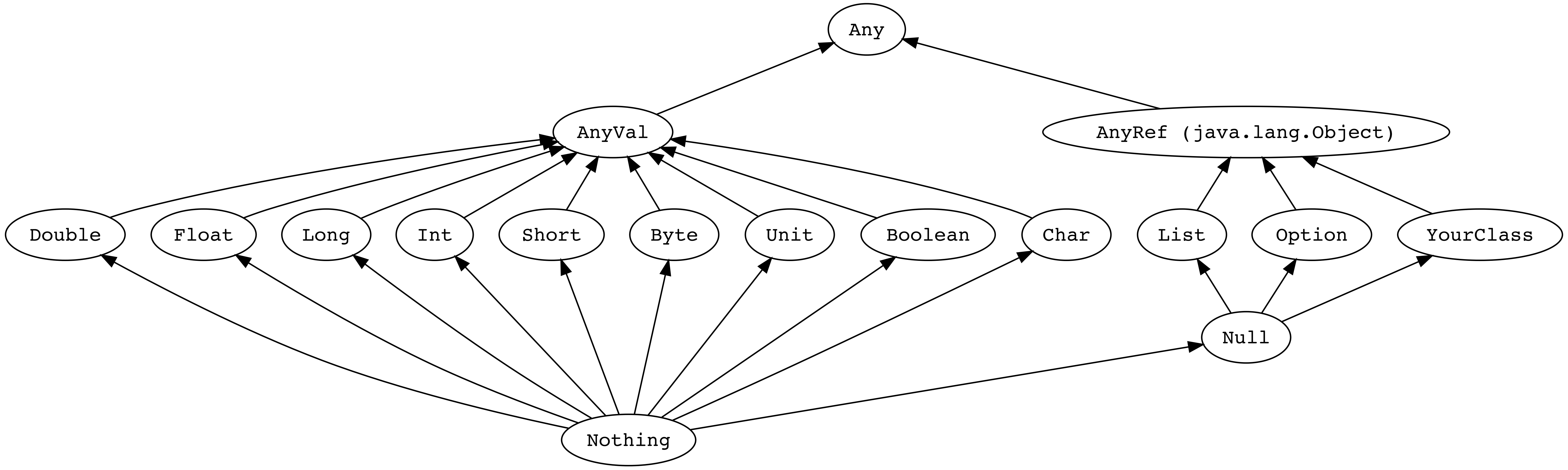
Inheritance

Override

Scala Type Hierarchy

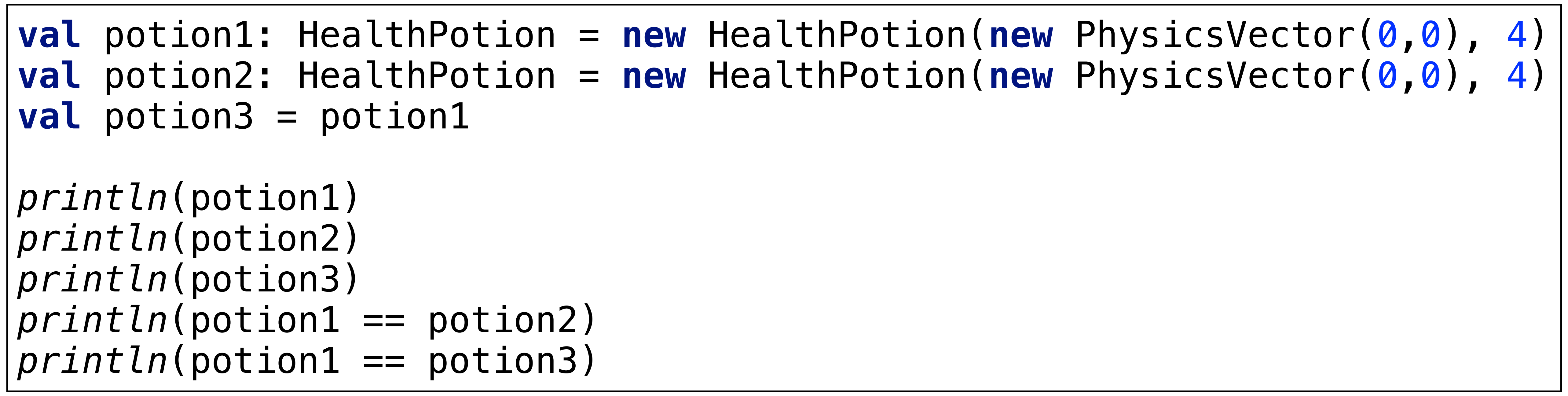


- All objects share Any as their base types
- Classes extending **AnyVal** will be stored on the **stack**
- Classes extending **AnyRef** will be stored on the **heap**

https://docs.scala-lang.org/tour/unified-types.html

Override

- Functionality is inherited from Any and AnyRef
- println calls an inherited .toString method
    - Converts object to a String with <object_type>@<reference>
- == calls the inherited .equals method
    - returns true only if the two variables refer to the same object in memory



```scala
val potion1: HealthPotion = new HealthPotion(new PhysicsVector(0,0), 4)
val potion2: HealthPotion = new HealthPotion(new PhysicsVector(0,0), 4)
val potion3 = potion1

println(potion1)
println(potion2)
println(potion3)
println(potion1 == potion2)
println(potion1 == potion3)
```

```
lo2_oop.oop_physics.with_oop.HealthPotion@17c68925
lo2_oop.oop_physics.with_oop.HealthPotion@7e0ea639
lo2_oop.oop_physics.with_oop.HealthPotion@17c68925
false
true
```

Override

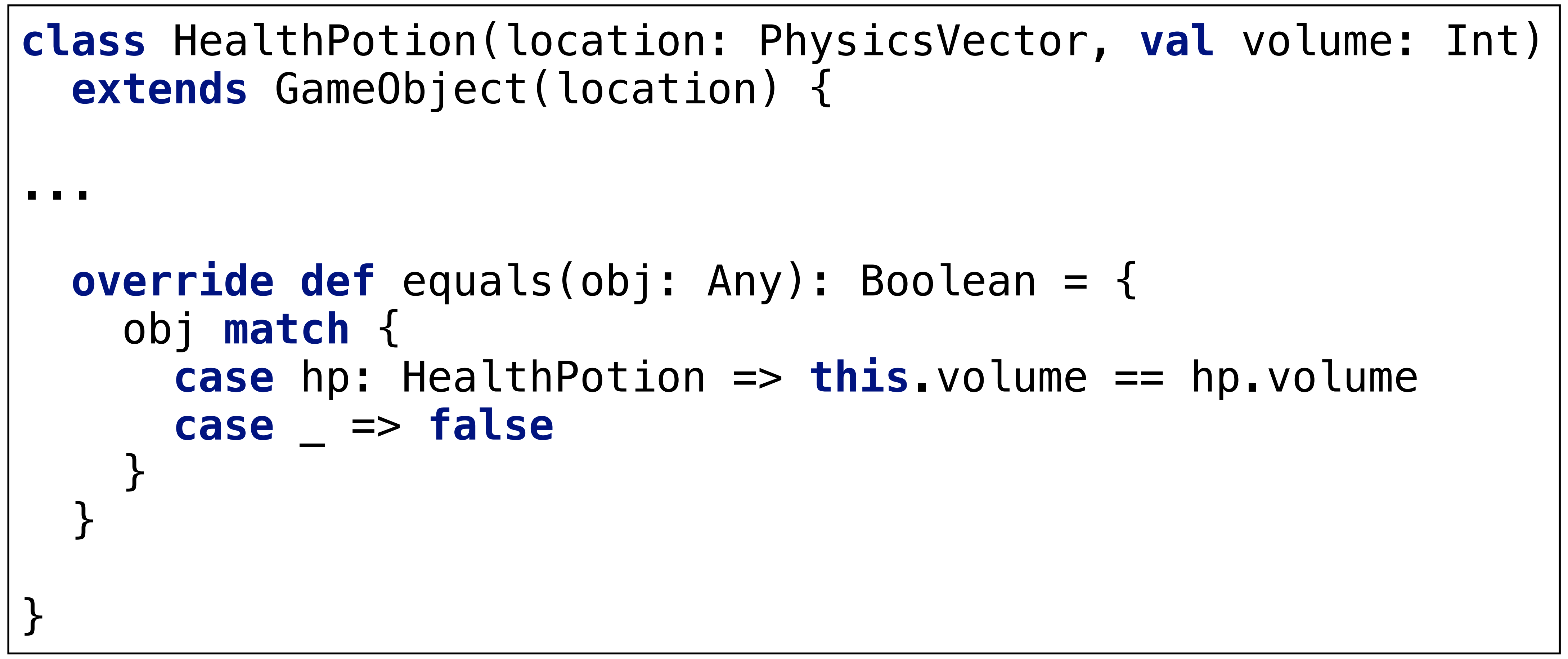
- We can override this default functionality
- Override toString to return a different string

```
class HealthPotion(location: PhysicsVector, val volume: Int)
  extends GameObject(location) {

...

  override def toString: String = {
    "location: " + this.location + "; volume: " + volume
  }

}
```

```
class PhysicsVector(var x: Double, var y: Double, var z: Double) {

  override def toString: String = {
    "(" + x + ", " + y + ", " + z + ")"
  }

}
```

Override

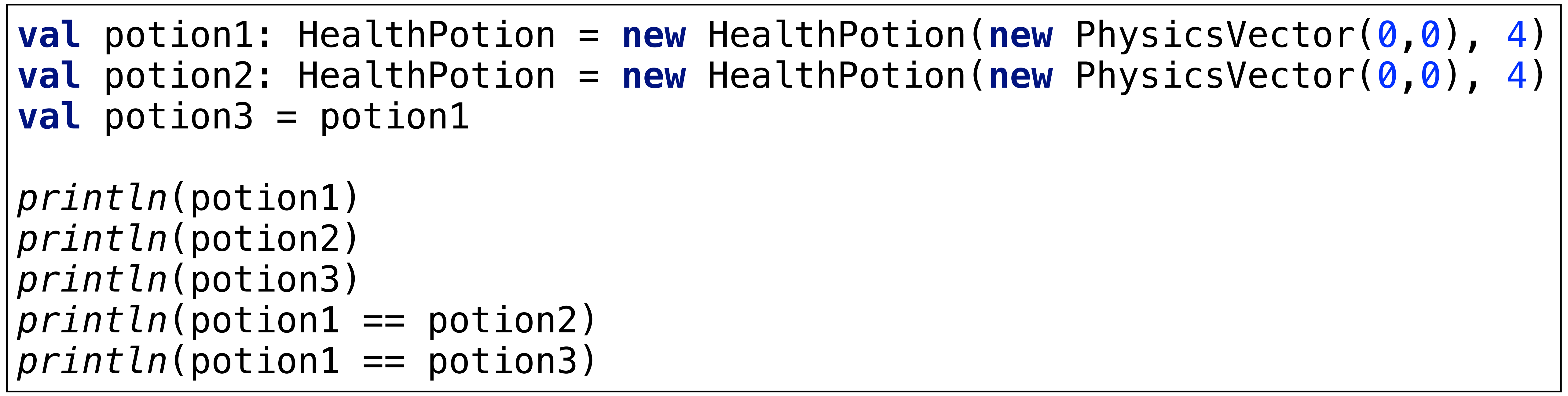
- Override equals to change the definition of equality
- Takes Any as a parameter
- Use match and case to behave differently on different types
- The _ wildcard covers all types not explicitly mentioned
- This method returns true when compared to another potion with the same volume, false otherwise



```
class HealthPotion(location: PhysicsVector, val volume: Int)
  extends GameObject(location) {

...

  override def equals(obj: Any): Boolean = {
    obj match {
      case hp: HealthPotion => this.volume == hp.volume
      case _ => false
    }
  }

}
```

Override

- With our overridden methods this code gives very different output



```
val potion1: HealthPotion = new HealthPotion(new PhysicsVector(0,0), 4)
val potion2: HealthPotion = new HealthPotion(new PhysicsVector(0,0), 4)
val potion3 = potion1

println(potion1)
println(potion2)
println(potion3)
println(potion1 == potion2)
println(potion1 == potion3)
```

```
location: (0.0, 0.0); volume: 4
location: (0.0, 0.0); volume: 4
location: (0.0, 0.0); volume: 4
true
true
```

Incoming Memory Diagram!!

```
abstract class PhysicsObject(var x: Double, var y: Double) {}
```

```
abstract class GameObject(var xObj: Double, var yObj: Double)
  extends PhysicsObject(xObj, yObj) {

  def objectMass(): Double

  override def toString: String = {
    "(" + this.x + ", " + this.y + "); mass: " + this.objectMass()
  }
}
```

```
class DodgeBall(var xDB: Double, var yDB: Double, val mass: Double)
  extends GameObject(xDB, yDB) {

  override def objectMass(): Double = {
    this.mass
  }
}
```

```
class HealthPotion(var xPotion: Double, var yPotion: Double,
                   val volume: Int)
  extends GameObject(xPotion, yPotion) {

  override def objectMass(): Double = {
    val massPerVolume: Double = 7.0
    this.volume * massPerVolume
  }

  override def toString: String = {
    "(" + this.x + ", " + this.y + "); volume: " + this.volume
  }
}
```

```
def main(args: Array[String]): Unit = {

  val ball: DodgeBall = new DodgeBall(-2.2, 4.8, 2)

  val potion1: HealthPotion = new HealthPotion(5.0, -3.5, 6)
  val potion2: HealthPotion = potion1

  ball.x += 1.0

  println(ball.objectMass())
  println(potion2.objectMass())
  println(ball.toString())
  println(potion1.toString())
}
```

```
abstract class PhysicsObject(var x: Double, var y: Double) {}
```

```
abstract class GameObject(var xObj: Double, var yObj: Double)
  extends PhysicsObject(xObj, yObj) {

  def objectMass(): Double

  override def toString: String = {
    "(" + this.x + ", " + this.y + "); mass: " + this.objectMass()
  }
}
```

```
class DodgeBall(var xDB: Double, var yDB: Double, val mass: Double)
  extends GameObject(xDB, yDB) {

  override def objectMass(): Double = {
    this.mass
  }
}
```

```
class HealthPotion(var xPotion: Double, var yPotion: Double,
                   val volume: Int)
  extends GameObject(xPotion, yPotion) {

  override def objectMass(): Double = {
    val massPerVolume: Double = 7.0
    this.volume * massPerVolume
  }

  override def toString: String = {
    "(" + this.x + ", " + this.y + "); volume: " + this.volume
  }
}
```

```
def main(args: Array[String]): Unit = {

  val ball: DodgeBall = new DodgeBall(-2.2, 4.8, 2)
  val potion1: HealthPotion = new HealthPotion(5.0, -3.5, 6)
  val potion2: HealthPotion = potion1

  ball.x += 1.0

  println(ball.objectMass())
  println(potion2.objectMass())
  println(ball.toString())
  println(potion1.toString())
}
```

- Let's start where it always begins
- The main method!

Stack

Name	Value

Heap

in/out

```
abstract class PhysicsObject(var x: Double, var y: Double) {}
```

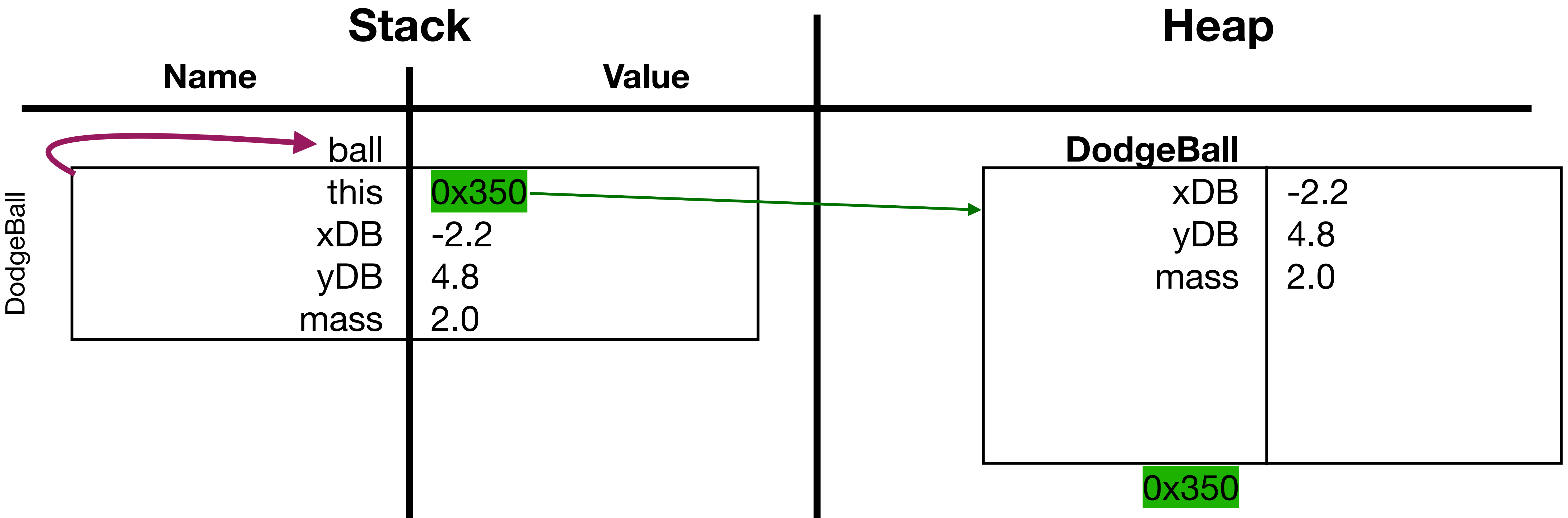
```
abstract class GameObject(var xObj: Double, var yObj: Double)
  extends PhysicsObject(xObj, yObj) {

  def objectMass(): Double

  override def toString: String = {
    "(" + this.x + ", " + this.y + "); mass: " + this.objectMass()
  }
}
```

```
class DodgeBall(var xDB: Double, var yDB: Double, val mass: Double)
  extends GameObject(xDB, yDB) {

  override def objectMass(): Double = {
    this.mass
  }
}
```

```
class HealthPotion(var xPotion: Double, var yPotion: Double,
                   val volume: Int)
  extends GameObject(xPotion, yPotion) {

  override def objectMass(): Double = {
    val massPerVolume: Double = 7.0
    this.volume * massPerVolume
  }

  override def toString: String = {
    "(" + this.x + ", " + this.y + "); volume: " + this.volume
  }
}
```

```
def main(args: Array[String]): Unit = {

  val ball: DodgeBall = new DodgeBall(-2.2, 4.8, 2)
  val potion1: HealthPotion = new HealthPotion(5.0, -3.5, 6)
  val potion2: HealthPotion = potion1

  ball.x += 1.0

  println(ball.objectMass())
  println(potion2.objectMass())
  println(ball.toString())
  println(potion1.toString())
}
```

- Create "ball" and and a **stack frame** for the call of the DodgeBall **constructor**
- Parameters become **state variables**



Stack

Name	Value
ball	
this	0x350
xDB	-2.2
yDB	4.8
mass	2.0

Heap

DodgeBall

xDB	-2.2
yDB	4.8
mass	2.0

0x350

in/out

```
abstract class PhysicsObject(var x: Double, var y: Double) {}
```

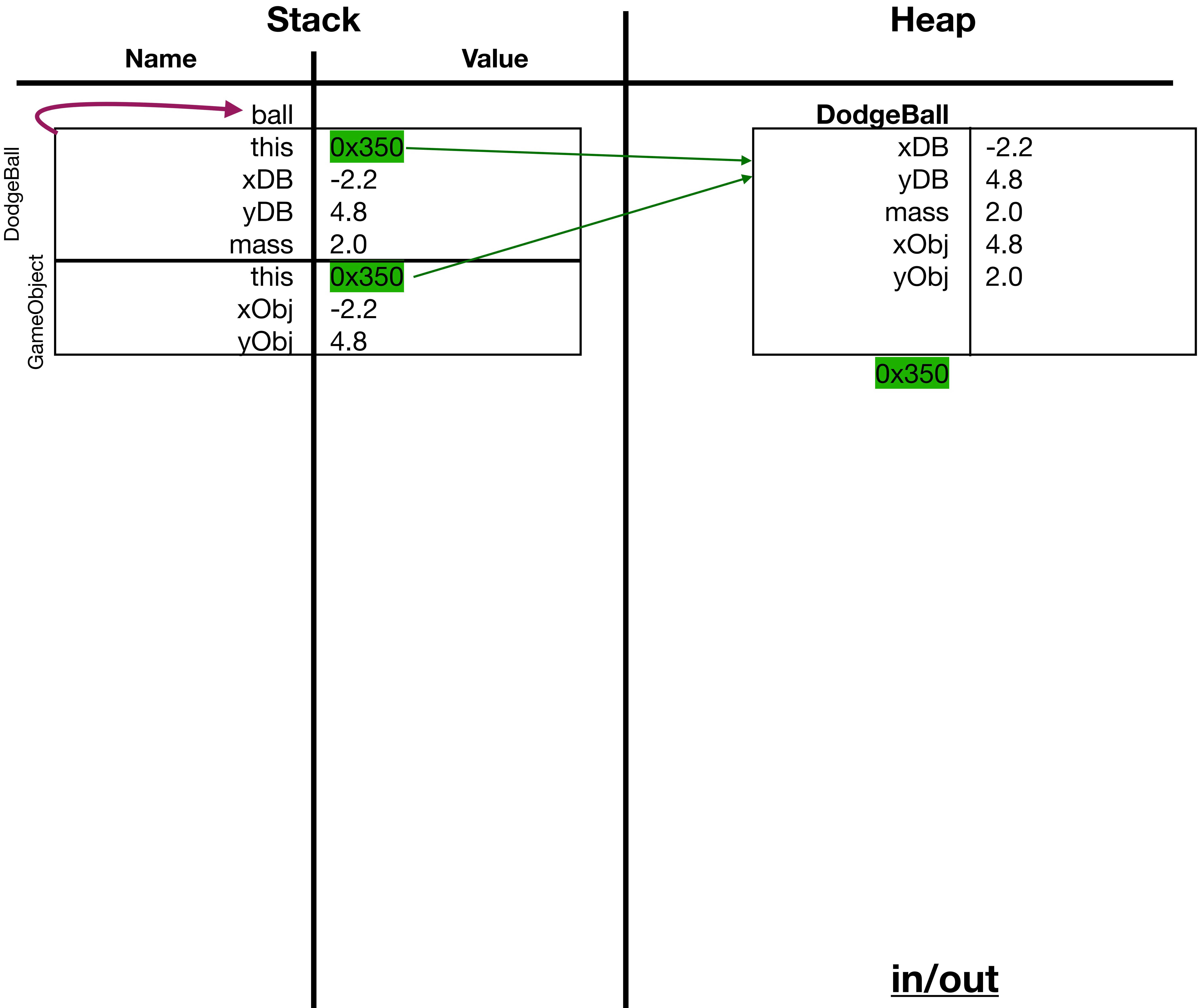
```
abstract class GameObject(var xObj: Double, var yObj: Double)
  extends PhysicsObject(xObj, yObj) {

  def objectMass(): Double

  override def toString: String = {
    "(" + this.x + ", " + this.y + "); mass: " + this.objectMass()
  }
}
```

```
class DodgeBall(var xDB: Double, var yDB: Double, val mass: Double)
  extends GameObject(xDB, yDB) {

  override def objectMass(): Double = {
    this.mass
  }
}
```

```
class HealthPotion(var xPotion: Double, var yPotion: Double,
                   val volume: Int)
  extends GameObject(xPotion, yPotion) {

  override def objectMass(): Double = {
    val massPerVolume: Double = 7.0
    this.volume * massPerVolume
  }

  override def toString: String = {
    "(" + this.x + ", " + this.y + "); volume: " + this.volume
  }
}
```

```
def main(args: Array[String]): Unit = {

  val ball: DodgeBall = new DodgeBall(-2.2, 4.8, 2)
  val potion1: HealthPotion = new HealthPotion(5.0, -3.5, 6)
  val potion2: HealthPotion = potion1

  ball.x += 1.0

  println(ball.objectMass())
  println(potion2.objectMass())
  println(ball.toString())
  println(potion1.toString())
}
```

- From the DodgeBall constructor, the GameObject constructor is called
- New stack frame; parameters become state variables



Name	Value
this	0x350
xDB	-2.2
yDB	4.8
mass	2.0
this	0x350
xObj	-2.2
yObj	4.8

DodgeBall	
xDB	-2.2
yDB	4.8
mass	2.0
xObj	4.8
yObj	2.0

```
abstract class PhysicsObject(var x: Double, var y: Double) {}
```

```
abstract class GameObject(var xObj: Double, var yObj: Double)
  extends PhysicsObject(xObj, yObj) {

  def objectMass(): Double

  override def toString: String = {
    "(" + this.x + ", " + this.y + "); mass: " + this.objectMass()
  }
}
```

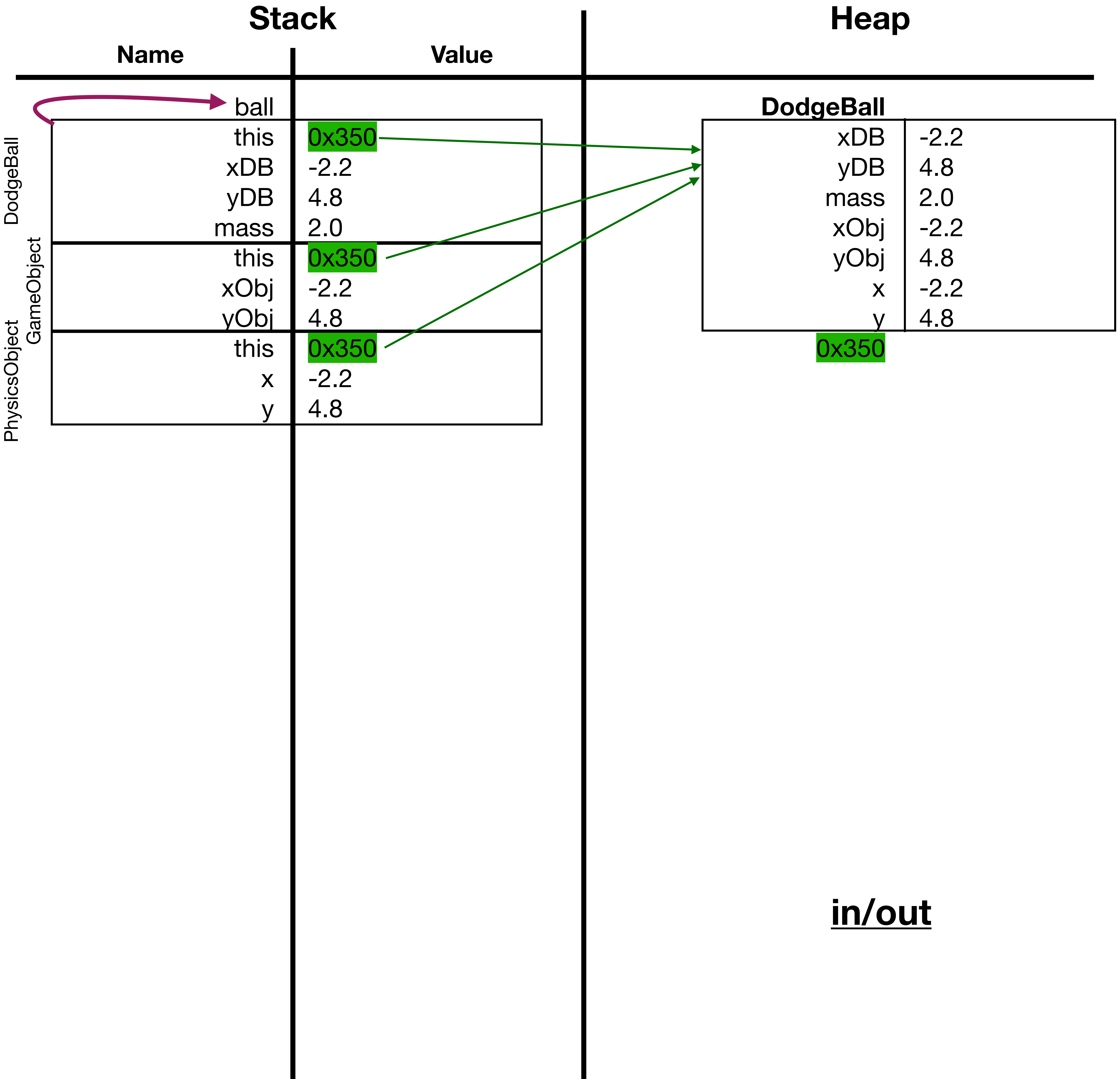
```
class DodgeBall(var xDB: Double, var yDB: Double, val mass: Double)
  extends GameObject(xDB, yDB) {

  override def objectMass(): Double = {
    this.mass
  }
}
```

```
class HealthPotion(var xPotion: Double, var yPotion: Double,
                   val volume: Int)
  extends GameObject(xPotion, yPotion) {

  override def objectMass(): Double = {
    val massPerVolume: Double = 7.0
    this.volume * massPerVolume
  }

  override def toString: String = {
    "(" + this.x + ", " + this.y + "); volume: " + this.volume
  }
}
```

```
def main(args: Array[String]): Unit = {

  val ball: DodgeBall = new DodgeBall(-2.2, 4.8, 2)
  val potion1: HealthPotion = new HealthPotion(5.0, -3.5, 6)
  val potion2: HealthPotion = potion1

  ball.x += 1.0

  println(ball.objectMass())
  println(potion2.objectMass())
  println(ball.toString())
  println(potion1.toString())
}
```

- GameObject constructor calls the PhysicsObject constructor

- Params from ALL 3 constructors become state variables



Stack

	Name	Value
	ball	
DodgeBall	this	0x350
	xDB	-2.2
	yDB	4.8
	mass	2.0
GameObject	this	0x350
	xObj	-2.2
	yObj	4.8
PhysicsObject	this	0x350
	x	-2.2
	y	4.8

Heap

DodgeBall

xDB	-2.2
yDB	4.8
mass	2.0
xObj	-2.2
yObj	4.8
x	-2.2
y	4.8

0x350

in/out

```
abstract class PhysicsObject(var x: Double, var y: Double) {}
```

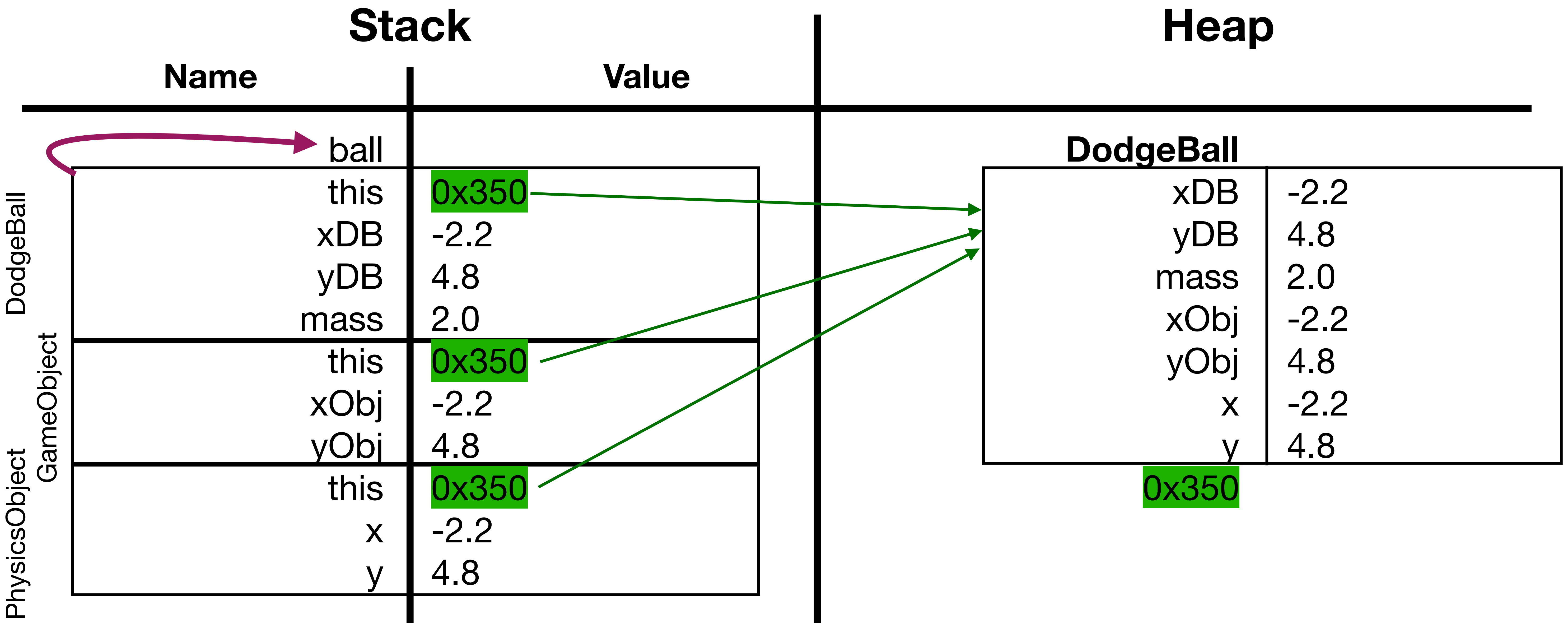
```
abstract class GameObject(var xObj: Double, var yObj: Double)
  extends PhysicsObject(xObj, yObj) {

  def objectMass(): Double

  override def toString: String = {
    "(" + this.x + ", " + this.y + "); mass: " + this.objectMass()
  }
}
```

```
class DodgeBall(var xDB: Double, var yDB: Double, val mass: Double)
  extends GameObject(xDB, yDB) {

  override def objectMass(): Double = {
    this.mass
  }
}
```

```
class HealthPotion(var xPotion: Double, var yPotion: Double,
                   val volume: Int)
  extends GameObject(xPotion, yPotion) {

  override def objectMass(): Double = {
    val massPerVolume: Double = 7.0
    this.volume * massPerVolume
  }

  override def toString: String = {
    "(" + this.x + ", " + this.y + "); volume: " + this.volume
  }
}
```

```
def main(args: Array[String]): Unit = {

  val ball: DodgeBall = new DodgeBall(-2.2, 4.8, 2)
  val potion1: HealthPotion = new HealthPotion(5.0, -3.5, 6)
  val potion2: HealthPotion = potion1

  ball.x += 1.0

  println(ball.objectMass())
  println(potion2.objectMass())
  println(ball.toString())
  println(potion1.toString())
}
```

- Parameters from **ALL 3** constructors become state variables!
- DodgeBall **inherits** these state variables from its **super classes**



Stack

	Name	Value
	ball	
DodgeBall	this	0x350
	xDB	-2.2
	yDB	4.8
	mass	2.0
GameObject	this	0x350
	xObj	-2.2
	yObj	4.8
PhysicsObject	this	0x350
	x	-2.2
	y	4.8

Heap

DodgeBall

xDB	-2.2
yDB	4.8
mass	2.0
xObj	-2.2
yObj	4.8
x	-2.2
y	4.8

0x350

in/out

```
abstract class PhysicsObject(var x: Double, var y: Double) {}
```

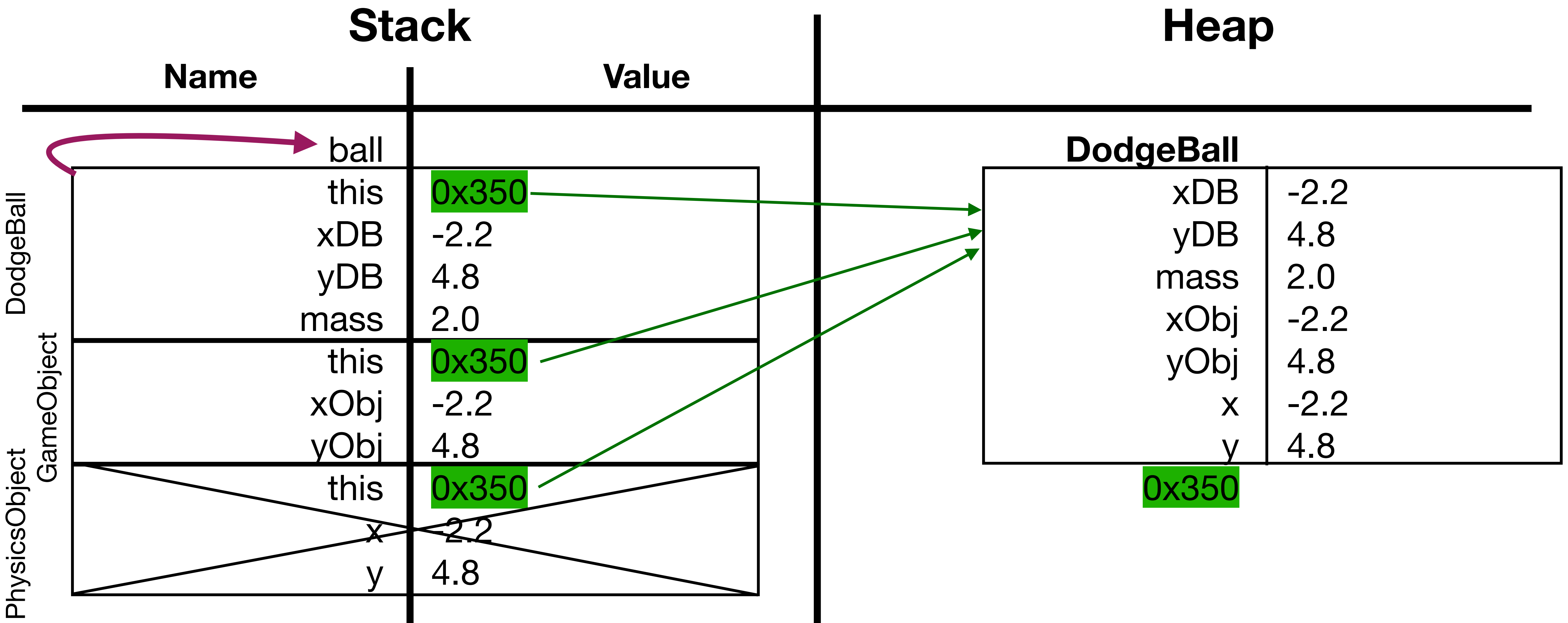
```
abstract class GameObject(var xObj: Double, var yObj: Double)
  extends PhysicsObject(xObj, yObj) {

  def objectMass(): Double

  override def toString: String = {
    "(" + this.x + ", " + this.y + "); mass: " + this.objectMass()
  }
}
```

```
class DodgeBall(var xDB: Double, var yDB: Double, val mass: Double)
  extends GameObject(xDB, yDB) {

  override def objectMass(): Double = {
    this.mass
  }
}
```

```
class HealthPotion(var xPotion: Double, var yPotion: Double,
                   val volume: Int)
  extends GameObject(xPotion, yPotion) {

  override def objectMass(): Double = {
    val massPerVolume: Double = 7.0
    this.volume * massPerVolume
  }

  override def toString: String = {
    "(" + this.x + ", " + this.y + "); volume: " + this.volume
  }
}
```

```
def main(args: Array[String]): Unit = {

  val ball: DodgeBall = new DodgeBall(-2.2, 4.8, 2)
  val potion1: HealthPotion = new HealthPotion(5.0, -3.5, 6)
  val potion2: HealthPotion = potion1

  ball.x += 1.0

  println(ball.objectMass())
  println(potion2.objectMass())
  println(ball.toString())
  println(potion1.toString())
}
```

- Run any code outside of the methods in PhysicsObject, then destroy the stack frame
  - [No code in this example]



Stack

	Name	Value
	ball	
DodgeBall	this	0x350
	xDB	-2.2
	yDB	4.8
	mass	2.0
GameObject	this	0x350
	xObj	-2.2
	yObj	4.8
PhysicsObject	~~this~~	~~0x350~~
	~~x~~	~~-2.2~~
	~~y~~	~~4.8~~

Heap

DodgeBall

xDB	-2.2
yDB	4.8
mass	2.0
xObj	-2.2
yObj	4.8
x	-2.2
y	4.8

0x350

in/out

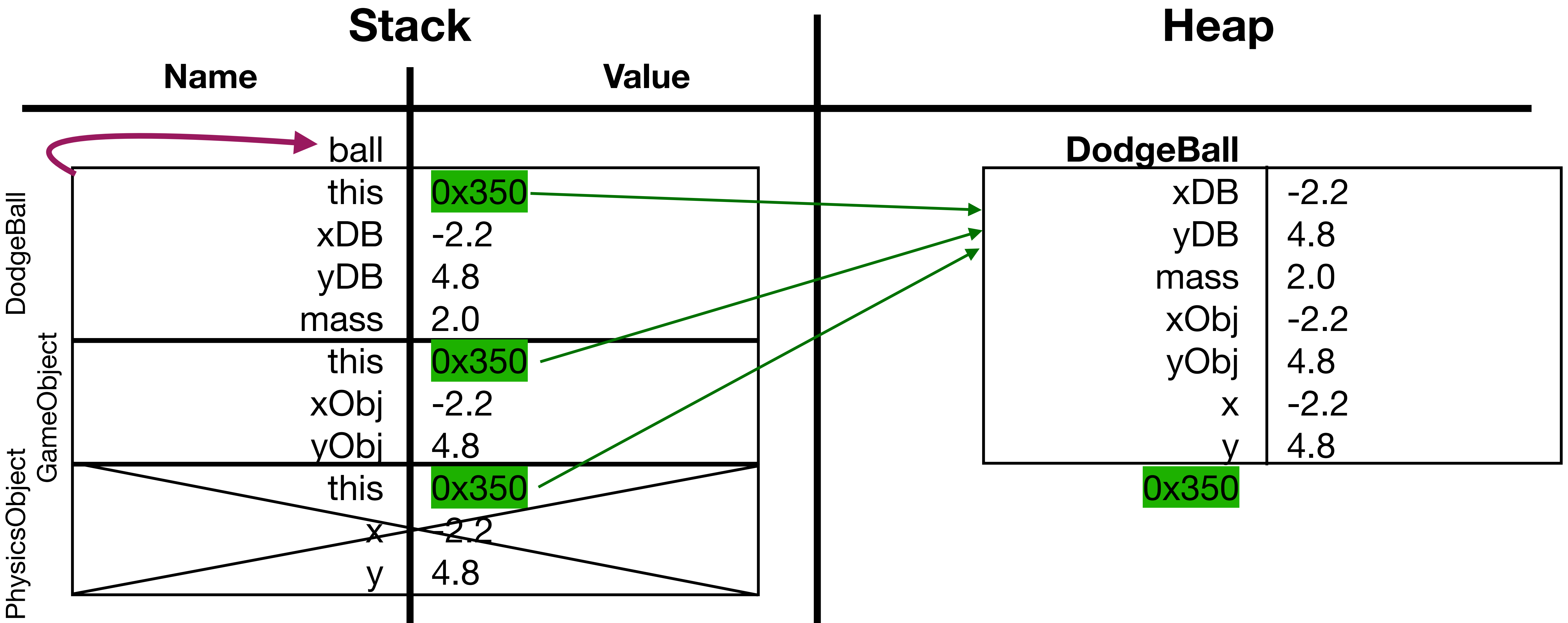
```
abstract class PhysicsObject(var x: Double, var y: Double) {}
```

```
abstract class GameObject(var xObj: Double, var yObj: Double)
  extends PhysicsObject(xObj, yObj) {

  def objectMass(): Double

  override def toString: String = {
    "(" + this.x + ", " + this.y + "); mass: " + this.objectMass()
  }
}
```

```
class DodgeBall(var xDB: Double, var yDB: Double, val mass: Double)
  extends GameObject(xDB, yDB) {

  override def objectMass(): Double = {
    this.mass
  }
}
```

```
class HealthPotion(var xPotion: Double, var yPotion: Double,
                   val volume: Int)
  extends GameObject(xPotion, yPotion) {

  override def objectMass(): Double = {
    val massPerVolume: Double = 7.0
    this.volume * massPerVolume
  }

  override def toString: String = {
    "(" + this.x + ", " + this.y + "); volume: " + this.volume
  }
}
```

```
def main(args: Array[String]): Unit = {

  val ball: DodgeBall = new DodgeBall(-2.2, 4.8, 2)
  val potion1: HealthPotion = new HealthPotion(5.0, -3.5, 6)
  val potion2: HealthPotion = potion1

  ball.x += 1.0

  println(ball.objectMass())
  println(potion2.objectMass())
  println(ball.toString())
  println(potion1.toString())
}
```



Stack

Name	Value
ball	
this	0x350
xDB	-2.2
yDB	4.8
mass	2.0
this	0x350
xObj	-2.2
yObj	4.8
this	0x350
~~x~~	~~-2.2~~
~~y~~	~~4.8~~

Heap

DodgeBall

xDB	-2.2
yDB	4.8
mass	2.0
xObj	-2.2
yObj	4.8
x	-2.2
y	4.8

0x350

in/out

- Run code outside methods in GameObject
  - [No code in this example]

```
abstract class PhysicsObject(var x: Double, var y: Double) {}
```

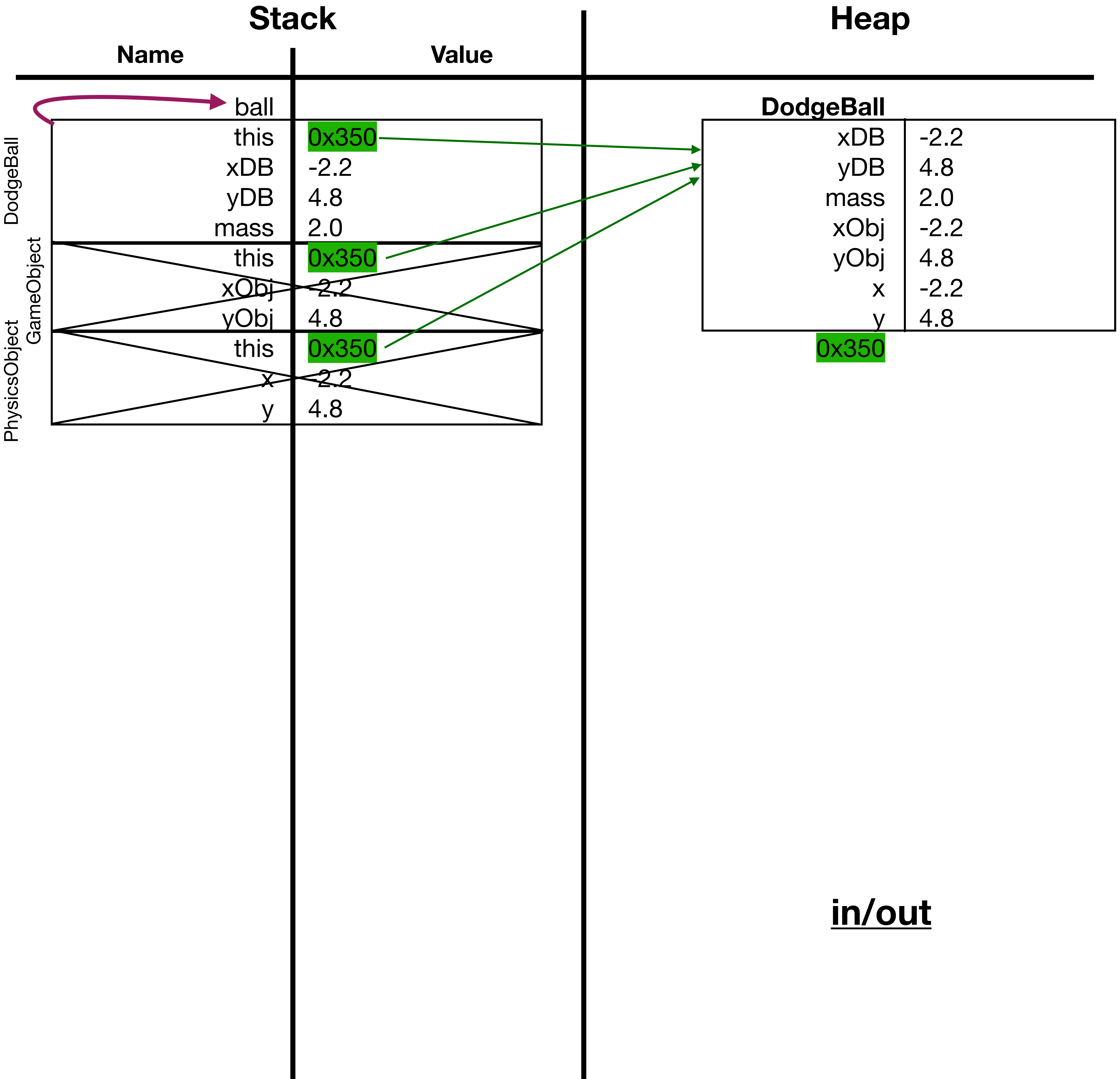
```
abstract class GameObject(var xObj: Double, var yObj: Double)
  extends PhysicsObject(xObj, yObj) {

  def objectMass(): Double

  override def toString: String = {
    "(" + this.x + ", " + this.y + "); mass: " + this.objectMass()
  }
}
```

```
class DodgeBall(var xDB: Double, var yDB: Double, val mass: Double)
  extends GameObject(xDB, yDB) {

  override def objectMass(): Double = {
    this.mass
  }
}
```

```
class HealthPotion(var xPotion: Double, var yPotion: Double,
                   val volume: Int)
  extends GameObject(xPotion, yPotion) {

  override def objectMass(): Double = {
    val massPerVolume: Double = 7.0
    this.volume * massPerVolume
  }

  override def toString: String = {
    "(" + this.x + ", " + this.y + "); volume: " + this.volume
  }
}
```

```
def main(args: Array[String]): Unit = {

  val ball: DodgeBall = new DodgeBall(-2.2, 4.8, 2)
  val potion1: HealthPotion = new HealthPotion(5.0, -3.5, 6)
  val potion2: HealthPotion = potion1

  ball.x += 1.0

  println(ball.objectMass())
  println(potion2.objectMass())
  println(ball.toString())
  println(potion1.toString())
}
```

- Repeat for DodgeBall



Stack

	Name	Value
	ball	
DodgeBall	this	0x350
	xDB	-2.2
	yDB	4.8
	mass	2.0
~~GameObject~~	~~this~~	~~0x350~~
	~~xObj~~	~~-2.2~~
	~~yObj~~	~~4.8~~
~~PhysicsObject~~	~~this~~	~~0x350~~
	~~x~~	~~-2.2~~
	~~y~~	~~4.8~~

Heap

DodgeBall

xDB	-2.2
yDB	4.8
mass	2.0
xObj	-2.2
yObj	4.8
x	-2.2
y	4.8

0x350

in/out

```
abstract class PhysicsObject(var x: Double, var y: Double) {}
```

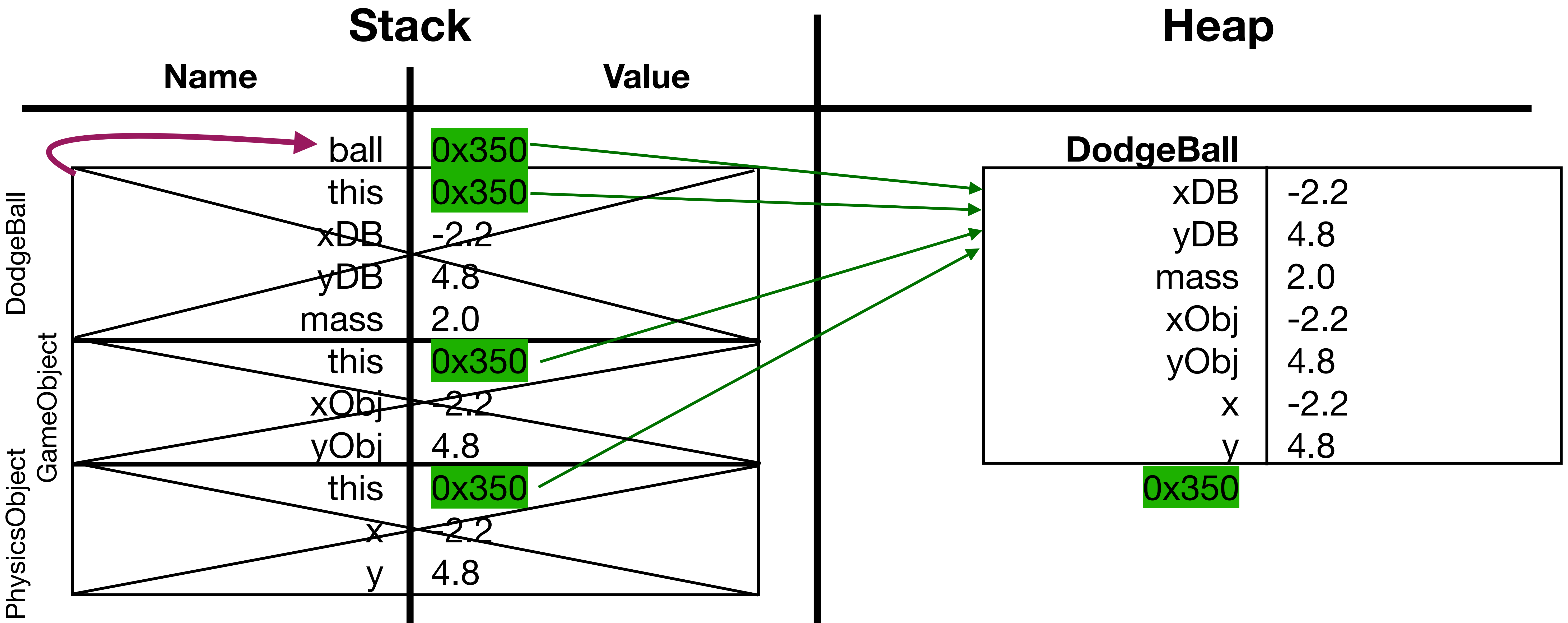
```
abstract class GameObject(var xObj: Double, var yObj: Double)
  extends PhysicsObject(xObj, yObj) {

  def objectMass(): Double

  override def toString: String = {
    "(" + this.x + ", " + this.y + "); mass: " + this.objectMass()
  }
}
```

```
class DodgeBall(var xDB: Double, var yDB: Double, val mass: Double)
  extends GameObject(xDB, yDB) {

  override def objectMass(): Double = {
    this.mass
  }
}
```

```
class HealthPotion(var xPotion: Double, var yPotion: Double,
                   val volume: Int)
  extends GameObject(xPotion, yPotion) {

  override def objectMass(): Double = {
    val massPerVolume: Double = 7.0
    this.volume * massPerVolume
  }

  override def toString: String = {
    "(" + this.x + ", " + this.y + "); volume: " + this.volume
  }
}
```

```
def main(args: Array[String]): Unit = {

  val ball: DodgeBall = new DodgeBall(-2.2, 4.8, 2)
  val potion1: HealthPotion = new HealthPotion(5.0, -3.5, 6)
  val potion2: HealthPotion = potion1

  ball.x += 1.0

  println(ball.objectMass())
  println(potion2.objectMass())
  println(ball.toString())
  println(potion1.toString())
}
```

- Stack frame for the DodgeBall constructor returns a reference to the object that was created



Stack

	Name	Value
	ball	0x350
DodgeBall	this	0x350
	xDB	-2.2
	yDB	4.8
	mass	2.0
GameObject	this	0x350
	xObj	-2.2
	yObj	4.8
PhysicsObject	this	0x350
	x	-2.2
	y	4.8

Heap

DodgeBall

xDB	-2.2
yDB	4.8
mass	2.0
xObj	-2.2
yObj	4.8
x	-2.2
y	4.8

0x350

in/out

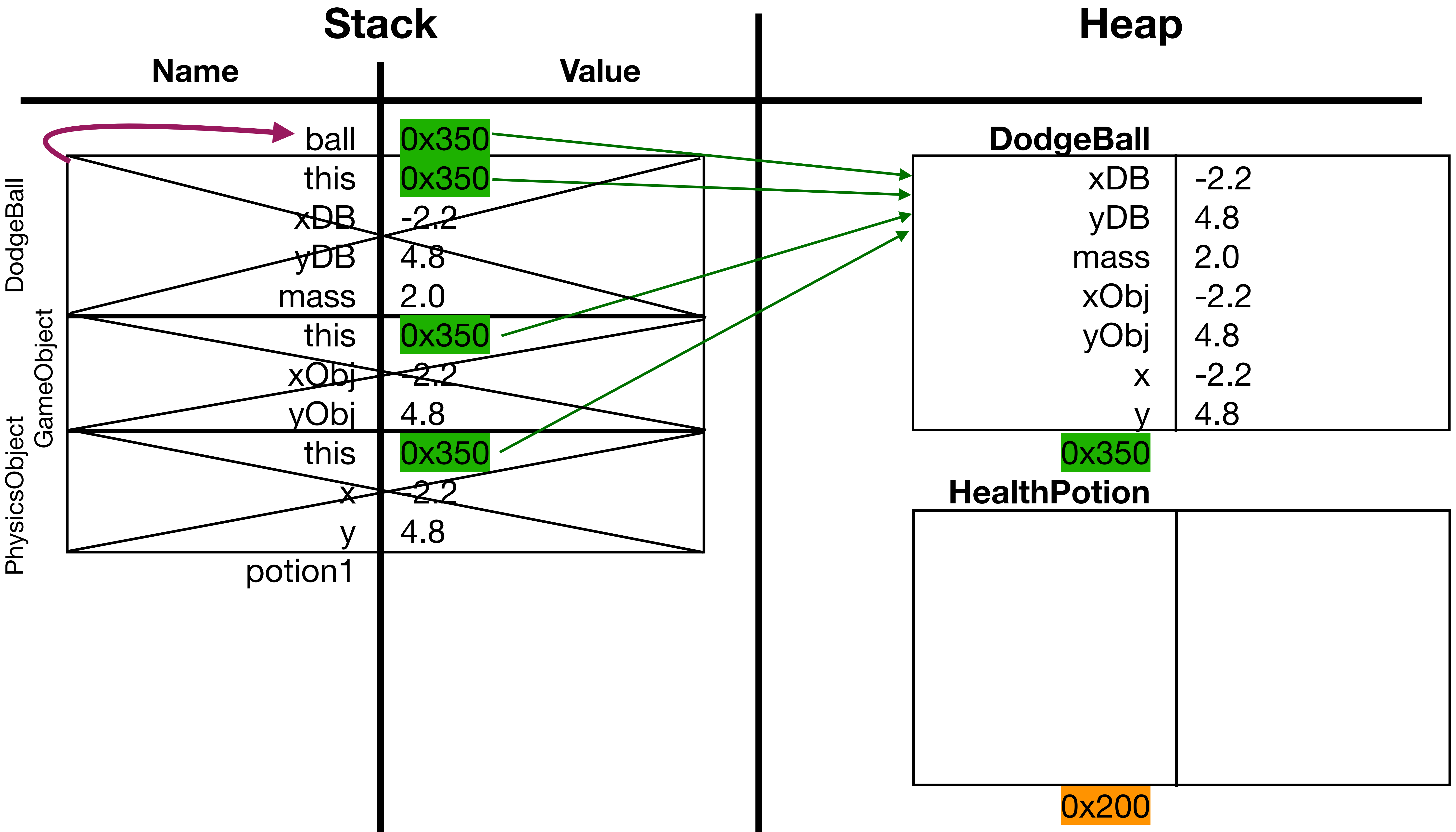
```
abstract class PhysicsObject(var x: Double, var y: Double) {}
```

```
abstract class GameObject(var xObj: Double, var yObj: Double)
  extends PhysicsObject(xObj, yObj) {

  def objectMass(): Double

  override def toString: String = {
    "(" + this.x + ", " + this.y + "); mass: " + this.objectMass()
  }
}
```

```
class DodgeBall(var xDB: Double, var yDB: Double, val mass: Double)
  extends GameObject(xDB, yDB) {

  override def objectMass(): Double = {
    this.mass
  }
}
```

```
class HealthPotion(var xPotion: Double, var yPotion: Double,
                   val volume: Int)
  extends GameObject(xPotion, yPotion) {

  override def objectMass(): Double = {
    val massPerVolume: Double = 7.0
    this.volume * massPerVolume
  }

  override def toString: String = {
    "(" + this.x + ", " + this.y + "); volume: " + this.volume
  }
}
```

```
def main(args: Array[String]): Unit = {

  val ball: DodgeBall = new DodgeBall(-2.2, 4.8, 2)
  val potion1: HealthPotion = new HealthPotion(5.0, -3.5, 6)
  val potion2: HealthPotion = potion1

  ball.x += 1.0

  println(ball.objectMass())
  println(potion2.objectMass())
  println(ball.toString())
  println(potion1.toString())
}
```



Stack

Name	Value
ball	0x350
this	0x350
xDB	-2.2
yDB	4.8
mass	2.0
this	0x350
xObj	-2.2
yObj	4.8
this	0x350
x	-2.2
y	4.8
potion1	

Heap

DodgeBall

xDB	-2.2
yDB	4.8
mass	2.0
xObj	-2.2
yObj	4.8
x	-2.2
y	4.8

0x350

HealthPotion

0x200

in/out

- **Exercise: How is the HealthPotion constructed?**

```
abstract class PhysicsObject(var x: Double, var y: Double) {}
```

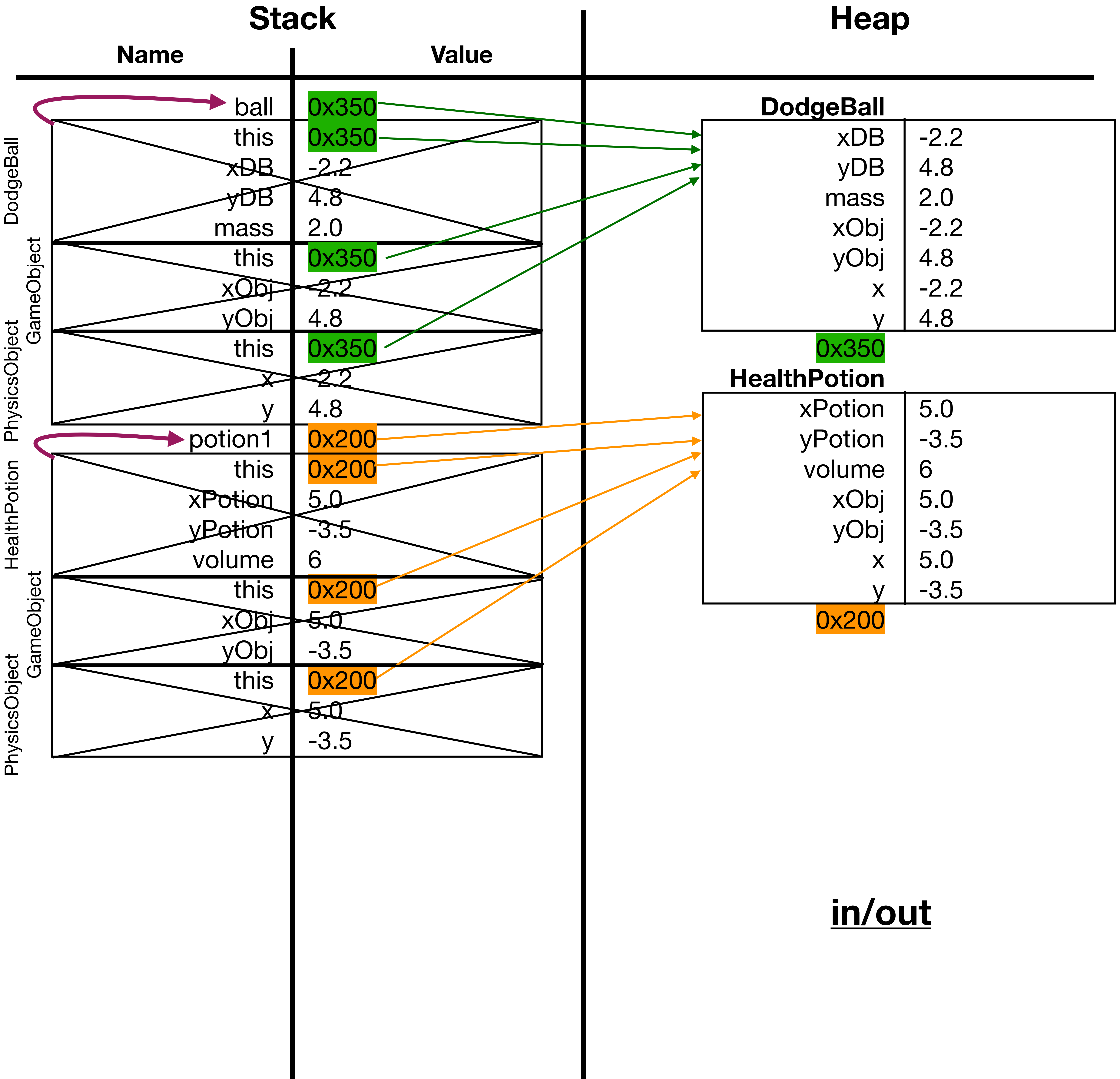
```
abstract class GameObject(var xObj: Double, var yObj: Double)
  extends PhysicsObject(xObj, yObj) {

  def objectMass(): Double

  override def toString: String = {
    "(" + this.x + ", " + this.y + "); mass: " + this.objectMass()
  }
}
```

```
class DodgeBall(var xDB: Double, var yDB: Double, val mass: Double)
  extends GameObject(xDB, yDB) {

  override def objectMass(): Double = {
    this.mass
  }
}
```

```
class HealthPotion(var xPotion: Double, var yPotion: Double,
                   val volume: Int)
  extends GameObject(xPotion, yPotion) {

  override def objectMass(): Double = {
    val massPerVolume: Double = 7.0
    this.volume * massPerVolume
  }

  override def toString: String = {
    "(" + this.x + ", " + this.y + "); volume: " + this.volume
  }
}
```

```
def main(args: Array[String]): Unit = {

  val ball: DodgeBall = new DodgeBall(-2.2, 4.8, 2)
  val potion1: HealthPotion = new HealthPotion(5.0, -3.5, 6)
  val potion2: HealthPotion = potion1

  ball.x += 1.0

  println(ball.objectMass())
  println(potion2.objectMass())
  println(ball.toString())
  println(potion1.toString())
}
```

- Exercise solution



Stack

	Name	Value
	ball	0x350
DodgeBall	this	0x350
	xDB	-2.2
	yDB	4.8
	mass	2.0
GameObject	this	0x350
	xObj	-2.2
	yObj	4.8
PhysicsObject	this	0x350
	x	-2.2
	y	4.8
	potion1	0x200
HealthPotion	this	0x200
	xPotion	5.0
	yPotion	-3.5
	volume	6
GameObject	this	0x200
	xObj	5.0
	yObj	-3.5
PhysicsObject	this	0x200
	x	5.0
	y	-3.5

Heap

DodgeBall

xDB	-2.2
yDB	4.8
mass	2.0
xObj	-2.2
yObj	4.8
x	-2.2
y	4.8

0x350

HealthPotion

xPotion	5.0
yPotion	-3.5
volume	6
xObj	5.0
yObj	-3.5
x	5.0
y	-3.5

0x200

in/out

```
abstract class PhysicsObject(var x: Double, var y: Double) {}
```

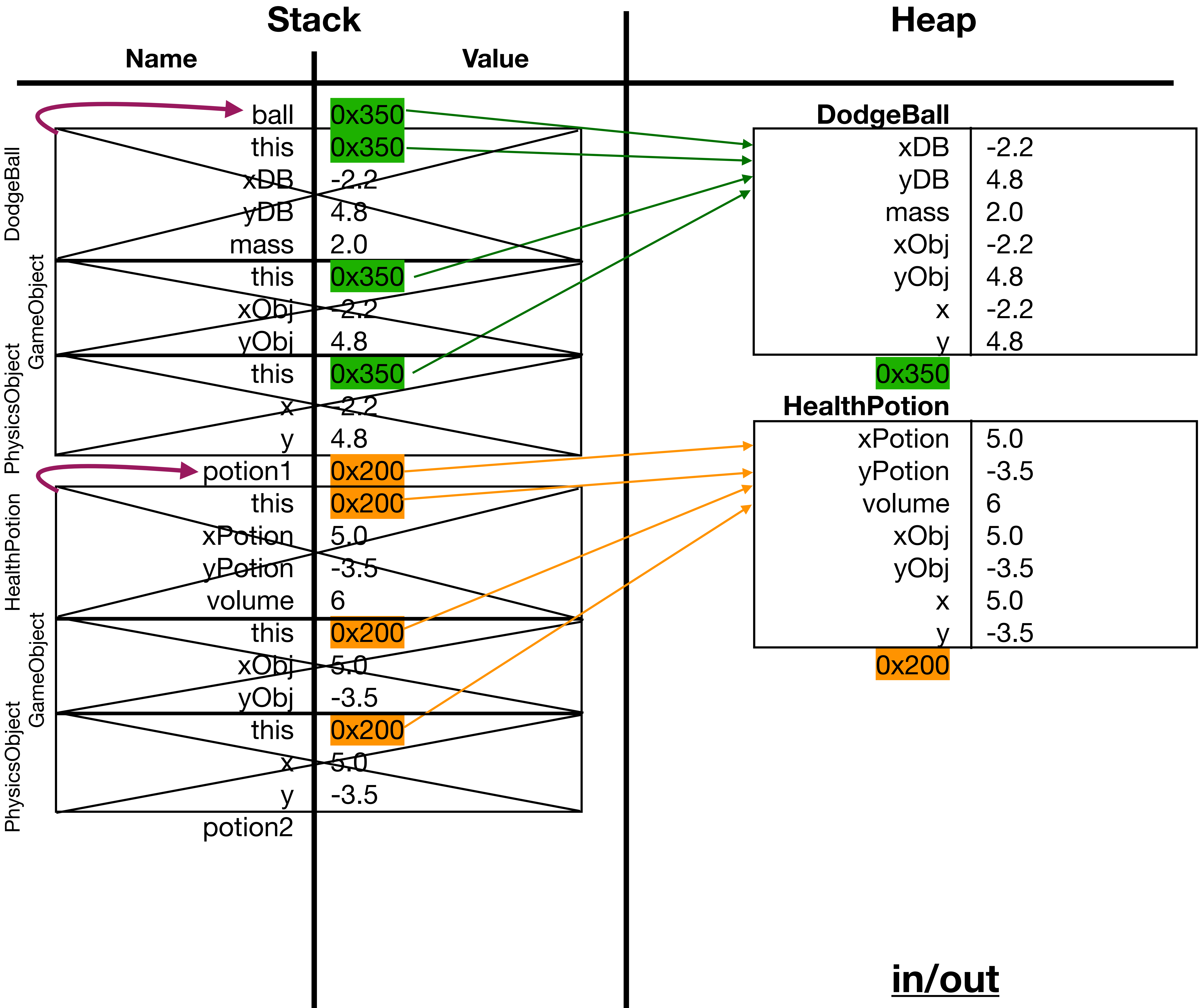
```
abstract class GameObject(var xObj: Double, var yObj: Double)
  extends PhysicsObject(xObj, yObj) {

  def objectMass(): Double

  override def toString: String = {
    "(" + this.x + ", " + this.y + "); mass: " + this.objectMass()
  }
}
```

```
class DodgeBall(var xDB: Double, var yDB: Double, val mass: Double)
  extends GameObject(xDB, yDB) {

  override def objectMass(): Double = {
    this.mass
  }
}
```

```
class HealthPotion(var xPotion: Double, var yPotion: Double,
                   val volume: Int)
  extends GameObject(xPotion, yPotion) {

  override def objectMass(): Double = {
    val massPerVolume: Double = 7.0
    this.volume * massPerVolume
  }

  override def toString: String = {
    "(" + this.x + ", " + this.y + "); volume: " + this.volume
  }
}
```

```
def main(args: Array[String]): Unit = {

  val ball: DodgeBall = new DodgeBall(-2.2, 4.8, 2)
  val potion1: HealthPotion = new HealthPotion(5.0, -3.5, 6)
  val potion2: HealthPotion = potion1

  ball.x += 1.0

  println(ball.objectMass())
  println(potion2.objectMass())
  println(ball.toString())
  println(potion1.toString())
}
```

- **Exercise: How about potion2?**



Stack

	Name	Value
	ball	0x350
DodgeBall	~~this~~	~~0x350~~
	~~xDB~~	~~-2.2~~
	~~yDB~~	~~4.8~~
	~~mass~~	~~2.0~~
GameObject	~~this~~	~~0x350~~
	~~xObj~~	~~-2.2~~
	~~yObj~~	~~4.8~~
PhysicsObject	~~this~~	~~0x350~~
	~~x~~	~~-2.2~~
	~~y~~	~~4.8~~
	potion1	0x200
HealthPotion	~~this~~	~~0x200~~
	~~xPotion~~	~~5.0~~
	~~yPotion~~	~~-3.5~~
	~~volume~~	~~6~~
GameObject	~~this~~	~~0x200~~
	~~xObj~~	~~5.0~~
	~~yObj~~	~~-3.5~~
PhysicsObject	~~this~~	~~0x200~~
	~~x~~	~~5.0~~
	~~y~~	~~-3.5~~
	potion2	

Heap

DodgeBall

xDB	-2.2
yDB	4.8
mass	2.0
xObj	-2.2
yObj	4.8
x	-2.2
y	4.8

0x350

HealthPotion

xPotion	5.0
yPotion	-3.5
volume	6
xObj	5.0
yObj	-3.5
x	5.0
y	-3.5

0x200

in/out

```scala
abstract class PhysicsObject(var x: Double, var y: Double) {}
```

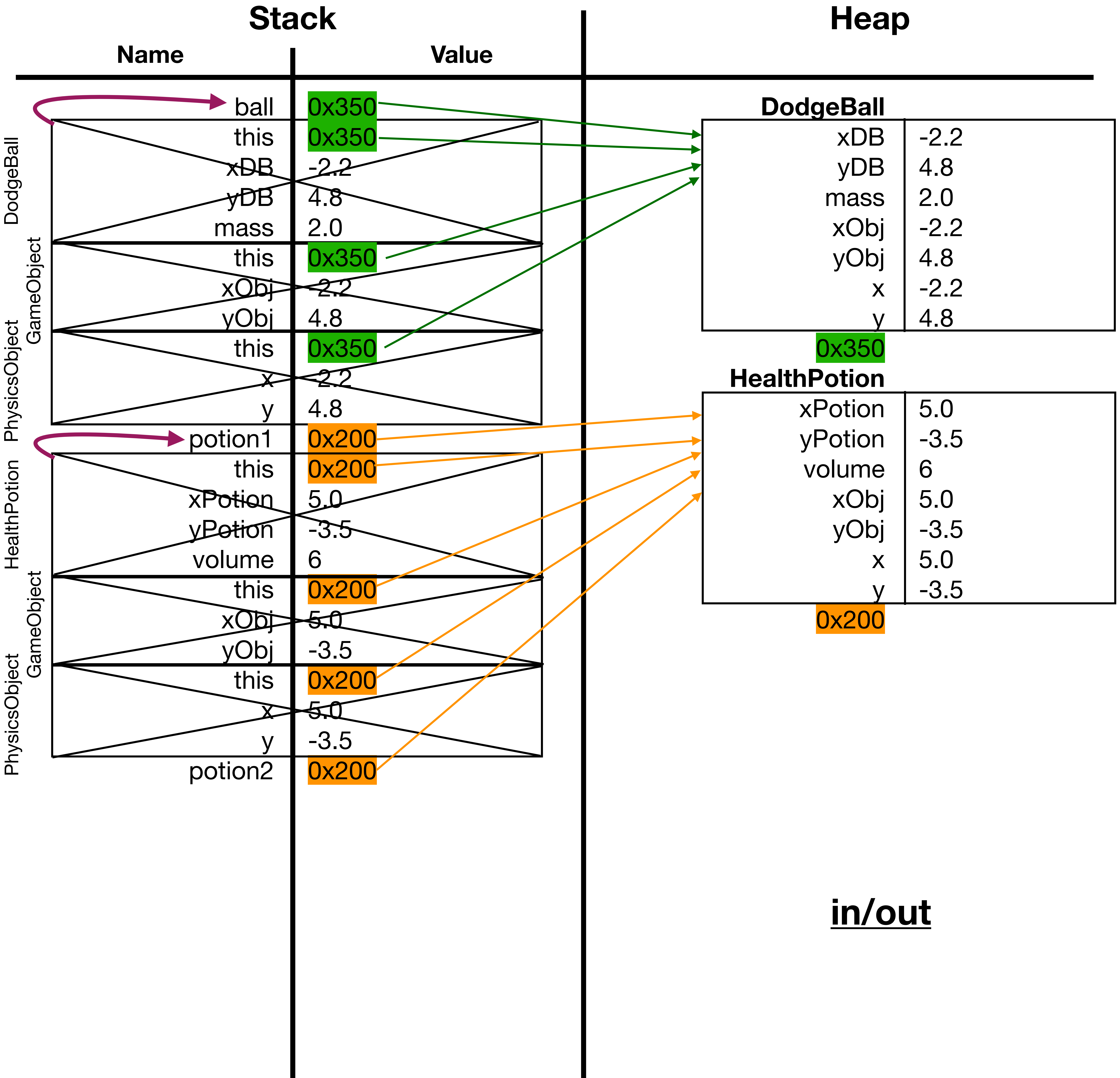
```scala
abstract class GameObject(var xObj: Double, var yObj: Double)
  extends PhysicsObject(xObj, yObj) {

  def objectMass(): Double

  override def toString: String = {
    "(" + this.x + ", " + this.y + "); mass: " + this.objectMass()
  }
}
```

```scala
class DodgeBall(var xDB: Double, var yDB: Double, val mass: Double)
  extends GameObject(xDB, yDB) {

  override def objectMass(): Double = {
    this.mass
  }
}
```

```scala
class HealthPotion(var xPotion: Double, var yPotion: Double,
                   val volume: Int)
  extends GameObject(xPotion, yPotion) {

  override def objectMass(): Double = {
    val massPerVolume: Double = 7.0
    this.volume * massPerVolume
  }

  override def toString: String = {
    "(" + this.x + ", " + this.y + "); volume: " + this.volume
  }
}
```

```scala
def main(args: Array[String]): Unit = {

  val ball: DodgeBall = new DodgeBall(-2.2, 4.8, 2)
  val potion1: HealthPotion = new HealthPotion(5.0, -3.5, 6)
  val potion2: HealthPotion = potion1

  ball.x += 1.0

  println(ball.objectMass())
  println(potion2.objectMass())
  println(ball.toString())
  println(potion1.toString())
}
```

- Exercise solution



Stack

	Name	Value
	ball	0x350
DodgeBall	this	0x350
	xDB	-2.2
	yDB	4.8
	mass	2.0
GameObject	this	0x350
	xObj	-2.2
	yObj	4.8
PhysicsObject	this	0x350
	x	-2.2
	y	4.8
	potion1	0x200
HealthPotion	this	0x200
	xPotion	5.0
	yPotion	-3.5
	volume	6
GameObject	this	0x200
	xObj	5.0
	yObj	-3.5
PhysicsObject	this	0x200
	x	5.0
	y	-3.5
	potion2	0x200

Heap

DodgeBall

xDB	-2.2
yDB	4.8
mass	2.0
xObj	-2.2
yObj	4.8
x	-2.2
y	4.8

0x350

HealthPotion

xPotion	5.0
yPotion	-3.5
volume	6
xObj	5.0
yObj	-3.5
x	5.0
y	-3.5

0x200

in/out

```
abstract class PhysicsObject(var x: Double, var y: Double) {}
```

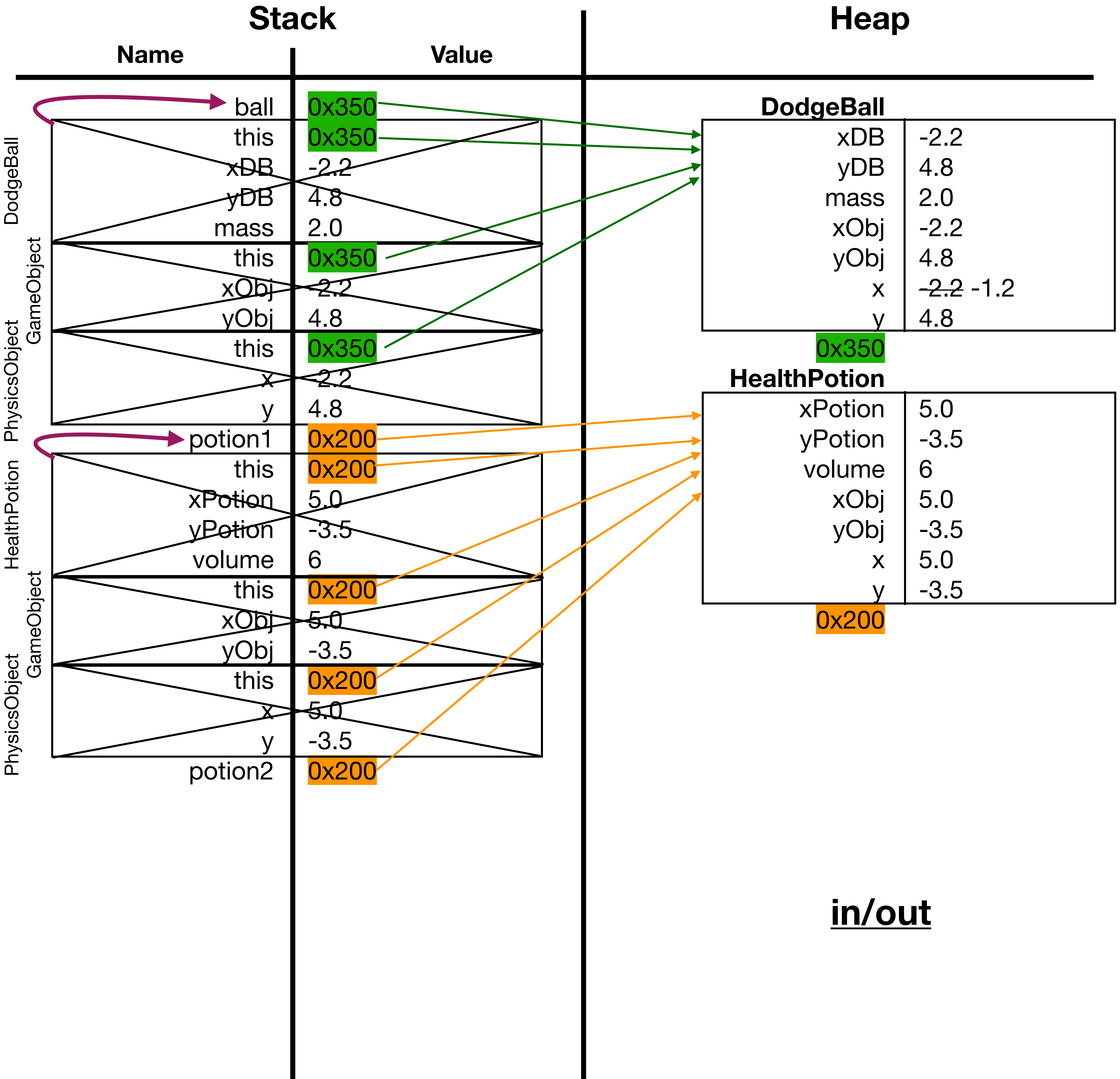
```
abstract class GameObject(var xObj: Double, var yObj: Double)
  extends PhysicsObject(xObj, yObj) {

  def objectMass(): Double

  override def toString: String = {
    "(" + this.x + ", " + this.y + "); mass: " + this.objectMass()
  }
}
```

```
class DodgeBall(var xDB: Double, var yDB: Double, val mass: Double)
  extends GameObject(xDB, yDB) {

  override def objectMass(): Double = {
    this.mass
  }
}
```

```
class HealthPotion(var xPotion: Double, var yPotion: Double,
                   val volume: Int)
  extends GameObject(xPotion, yPotion) {

  override def objectMass(): Double = {
    val massPerVolume: Double = 7.0
    this.volume * massPerVolume
  }

  override def toString: String = {
    "(" + this.x + ", " + this.y + "); volume: " + this.volume
  }
}
```

```
def main(args: Array[String]): Unit = {

  val ball: DodgeBall = new DodgeBall(-2.2, 4.8, 2)
  val potion1: HealthPotion = new HealthPotion(5.0, -3.5, 6)
  val potion2: HealthPotion = potion1

  ball.x += 1.0

  println(ball.objectMass())
  println(potion2.objectMass())
  println(ball.toString())
  println(potion1.toString())
}
```

- Update the x state variable of the DodgeBall



Stack

Name	Value
ball	0x350
this	0x350
xDB	-2.2
yDB	4.8
mass	2.0
this	0x350
xObj	-2.2
yObj	4.8
this	0x350
x	-2.2
y	4.8
potion1	0x200
this	0x200
xPotion	5.0
yPotion	-3.5
volume	6
this	0x200
xObj	5.0
yObj	-3.5
this	0x200
x	5.0
y	-3.5
potion2	0x200

Heap

DodgeBall

xDB	-2.2
yDB	4.8
mass	2.0
xObj	-2.2
yObj	4.8
x	~~-2.2~~ -1.2
y	4.8

0x350

HealthPotion

xPotion	5.0
yPotion	-3.5
volume	6
xObj	5.0
yObj	-3.5
x	5.0
y	-3.5

0x200

in/out

```
abstract class PhysicsObject(var x: Double, var y: Double) {}
```

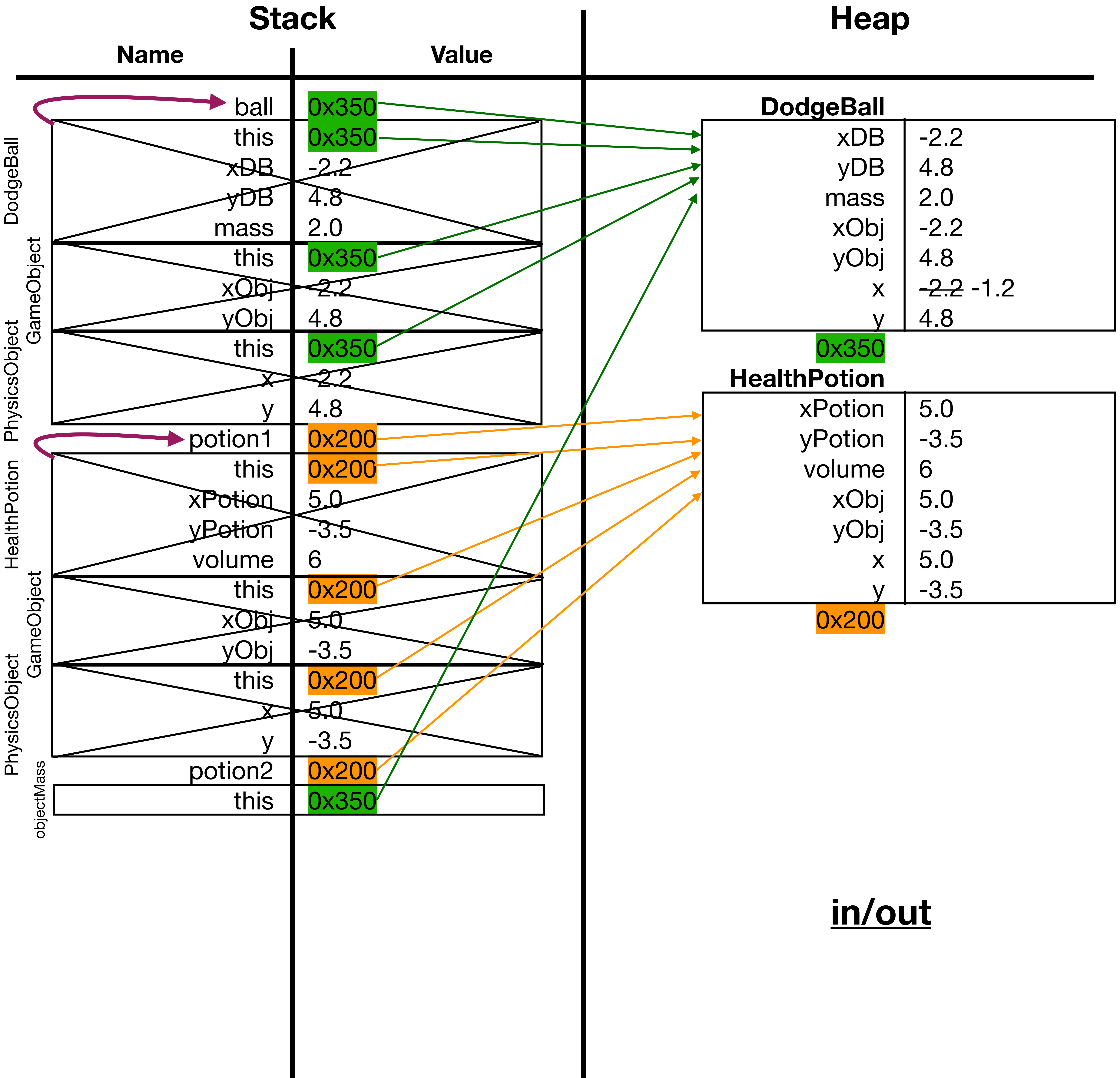
```
abstract class GameObject(var xObj: Double, var yObj: Double)
  extends PhysicsObject(xObj, yObj) {

  def objectMass(): Double

  override def toString: String = {
    "(" + this.x + ", " + this.y + "); mass: " + this.objectMass()
  }
}
```

```
class DodgeBall(var xDB: Double, var yDB: Double, val mass: Double)
  extends GameObject(xDB, yDB) {

  override def objectMass(): Double = {
    this.mass
  }
}
```

```
class HealthPotion(var xPotion: Double, var yPotion: Double,
                   val volume: Int)
  extends GameObject(xPotion, yPotion) {

  override def objectMass(): Double = {
    val massPerVolume: Double = 7.0
    this.volume * massPerVolume
  }

  override def toString: String = {
    "(" + this.x + ", " + this.y + "); volume: " + this.volume
  }
}
```

```
def main(args: Array[String]): Unit = {

  val ball: DodgeBall = new DodgeBall(-2.2, 4.8, 2)
  val potion1: HealthPotion = new HealthPotion(5.0, -3.5, 6)
  val potion2: HealthPotion = potion1

  ball.x += 1.0

  println(ball.objectMass())
  println(potion2.objectMass())
  println(ball.toString())
  println(potion1.toString())
}
```

- ball.objectMass()
- What objectMass method is called? Why?



Stack

	Name	Value
	ball	0x350
DodgeBall	this	0x350
	xDB	-2.2
	yDB	4.8
	mass	2.0
GameObject	this	0x350
	xObj	-2.2
	yObj	4.8
PhysicsObject	this	0x350
	x	-2.2
	y	4.8
	potion1	0x200
HealthPotion	this	0x200
	xPotion	5.0
	yPotion	-3.5
	volume	6
GameObject	this	0x200
	xObj	5.0
	yObj	-3.5
PhysicsObject	this	0x200
	x	5.0
	y	-3.5
	potion2	0x200
objectMass	this	0x350

Heap

DodgeBall

xDB	-2.2
yDB	4.8
mass	2.0
xObj	-2.2
yObj	4.8
x	~~-2.2~~ -1.2
y	4.8

0x350

HealthPotion

xPotion	5.0
yPotion	-3.5
volume	6
xObj	5.0
yObj	-3.5
x	5.0
y	-3.5

0x200

in/out

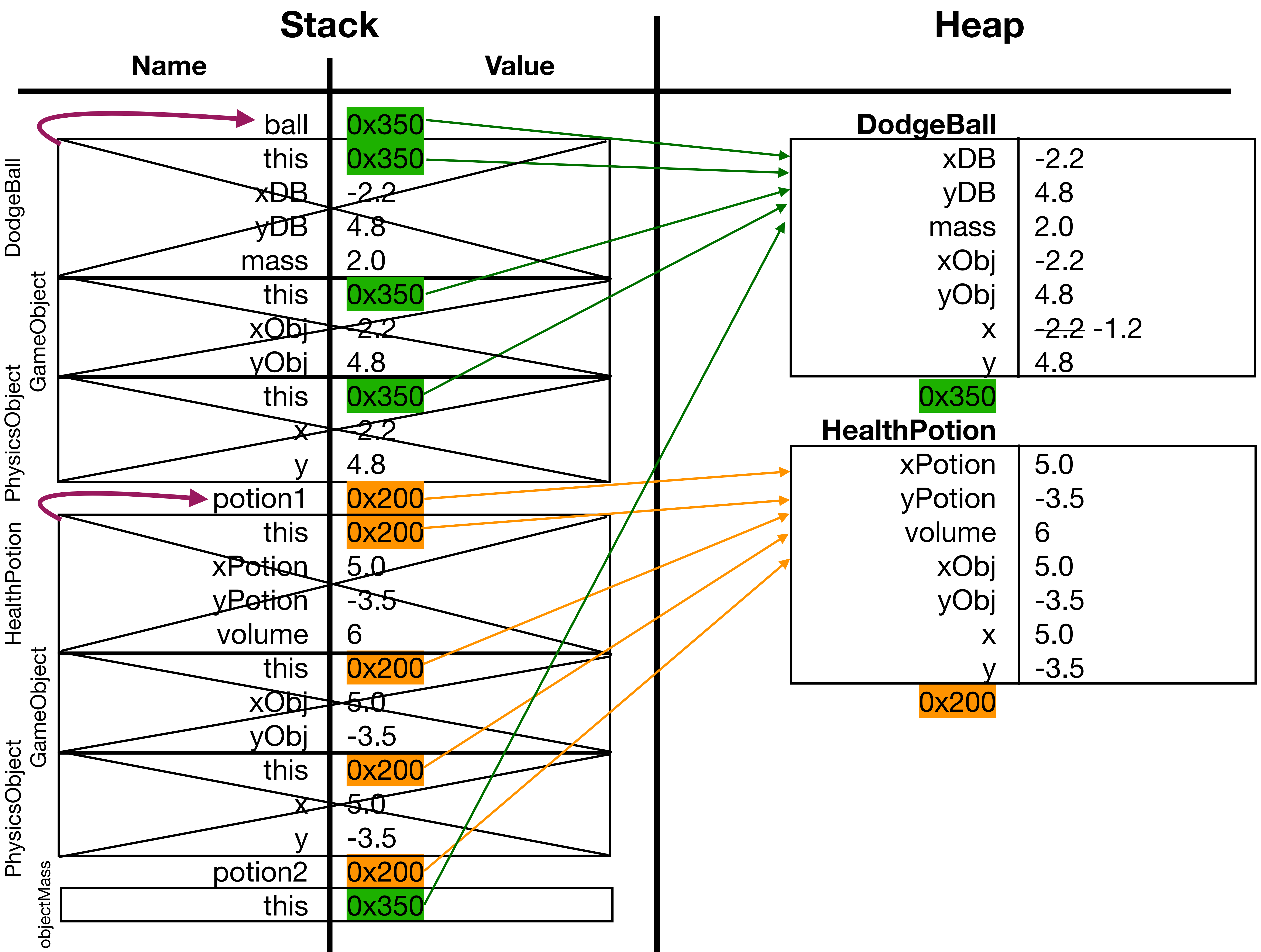
```
abstract class PhysicsObject(var x: Double, var y: Double) {}
```

```
abstract class GameObject(var xObj: Double, var yObj: Double)
  extends PhysicsObject(xObj, yObj) {

  def objectMass(): Double

  override def toString: String = {
    "(" + this.x + ", " + this.y + "); mass: " + this.objectMass()
  }
}
```

```
class DodgeBall(var xDB: Double, var yDB: Double, val mass: Double)
  extends GameObject(xDB, yDB) {

  override def objectMass(): Double = {
    this.mass
  }
}
```

```
class HealthPotion(var xPotion: Double, var yPotion: Double,
                   val volume: Int)
  extends GameObject(xPotion, yPotion) {

  override def objectMass(): Double = {
    val massPerVolume: Double = 7.0
    this.volume * massPerVolume
  }

  override def toString: String = {
    "(" + this.x + ", " + this.y + "); volume: " + this.volume
  }
}
```

```
def main(args: Array[String]): Unit = {

  val ball: DodgeBall = new DodgeBall(-2.2, 4.8, 2)
  val potion1: HealthPotion = new HealthPotion(5.0, -3.5, 6)
  val potion2: HealthPotion = potion1

  ball.x += 1.0

  println(ball.objectMass())
  println(potion2.objectMass())
  println(ball.toString())
  println(potion1.toString())
}
```



Stack

	Name	Value
	ball	0x350
DodgeBall	this	0x350
	xDB	-2.2
	yDB	4.8
	mass	2.0
GameObject	this	0x350
	xObj	-2.2
	yObj	4.8
PhysicsObject	this	0x350
	x	-2.2
	y	4.8
	potion1	0x200
HealthPotion	this	0x200
	xPotion	5.0
	yPotion	-3.5
	volume	6
GameObject	this	0x200
	xObj	5.0
	yObj	-3.5
PhysicsObject	this	0x200
	x	5.0
	y	-3.5
	potion2	0x200
objectMass	this	0x350

Heap

DodgeBall

xDB	-2.2
yDB	4.8
mass	2.0
xObj	-2.2
yObj	4.8
x	~~-2.2~~ -1.2
y	4.8

0x350

HealthPotion

xPotion	5.0
yPotion	-3.5
volume	6
xObj	5.0
yObj	-3.5
x	5.0
y	-3.5

0x200

in/out

- Follow the reference and check the **type** of the object on the heap
- ball stores 0x350 which refers to a **DodgeBall**

```
abstract class PhysicsObject(var x: Double, var y: Double) {}
```

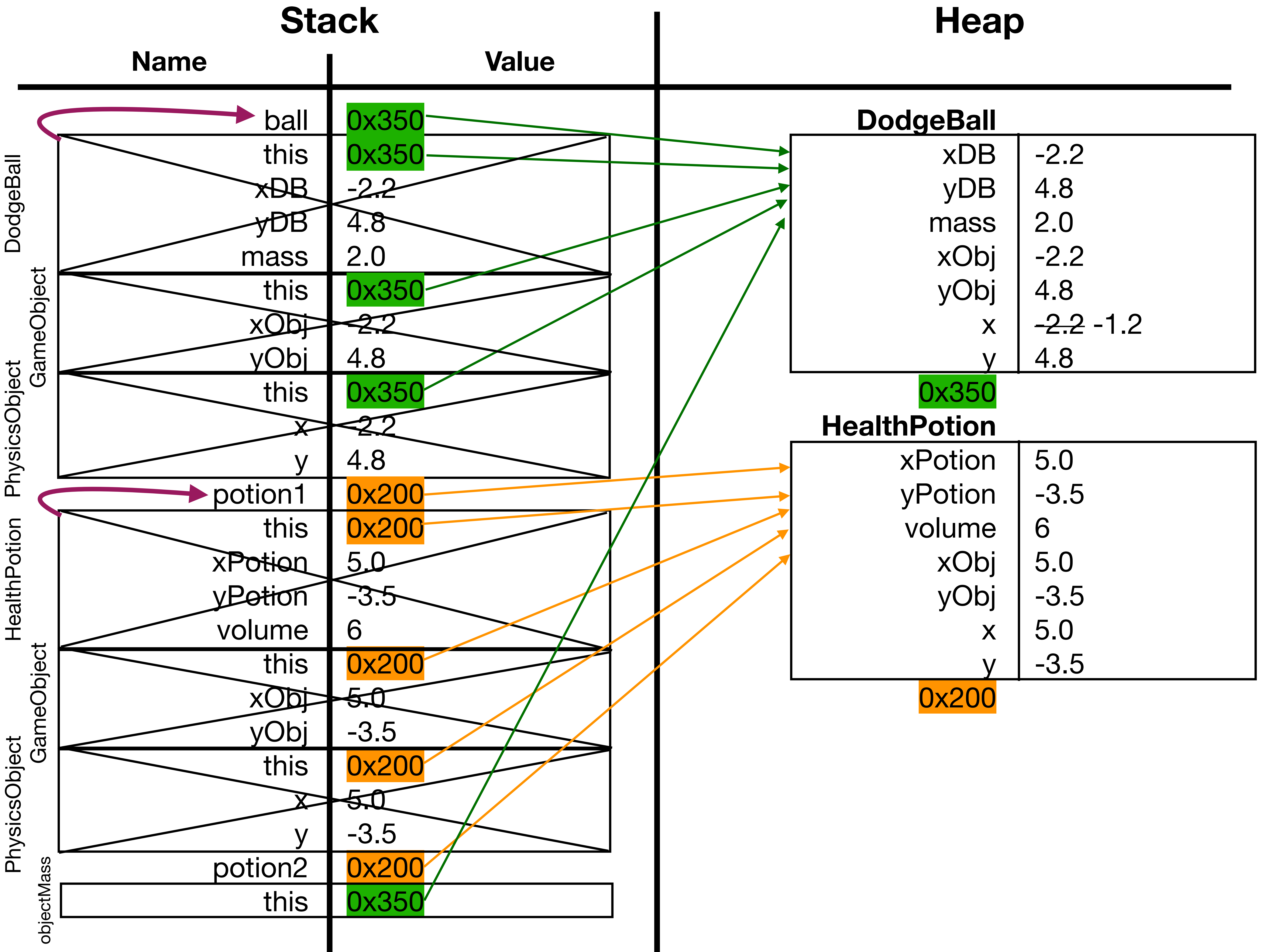
```
abstract class GameObject(var xObj: Double, var yObj: Double)
  extends PhysicsObject(xObj, yObj) {

  def objectMass(): Double

  override def toString: String = {
    "(" + this.x + ", " + this.y + "); mass: " + this.objectMass()
  }
}
```

```
class DodgeBall(var xDB: Double, var yDB: Double, val mass: Double)
  extends GameObject(xDB, yDB) {

  override def objectMass(): Double = {
    this.mass
  }
}
```

```
class HealthPotion(var xPotion: Double, var yPotion: Double,
                   val volume: Int)
  extends GameObject(xPotion, yPotion) {

  override def objectMass(): Double = {
    val massPerVolume: Double = 7.0
    this.volume * massPerVolume
  }

  override def toString: String = {
    "(" + this.x + ", " + this.y + "); volume: " + this.volume
  }
}
```

```
def main(args: Array[String]): Unit = {

  val ball: DodgeBall = new DodgeBall(-2.2, 4.8, 2)
  val potion1: HealthPotion = new HealthPotion(5.0, -3.5, 6)
  val potion2: HealthPotion = potion1

  ball.x += 1.0

  println(ball.objectMass())
  println(potion2.objectMass())
  println(ball.toString())
  println(potion1.toString())
}
```

- Call objectMass defined in DodgeBall
- Prints 2.0 to the screen



Stack

	Name	Value
	ball	0x350
DodgeBall	this	0x350
	xDB	-2.2
	yDB	4.8
	mass	2.0
GameObject	this	0x350
	xObj	-2.2
	yObj	4.8
PhysicsObject	this	0x350
	x	-2.2
	y	4.8
	potion1	0x200
HealthPotion	this	0x200
	xPotion	5.0
	yPotion	-3.5
	volume	6
GameObject	this	0x200
	xObj	5.0
	yObj	-3.5
PhysicsObject	this	0x200
	x	5.0
	y	-3.5
	potion2	0x200
objectMass	this	0x350

Heap

DodgeBall

xDB	-2.2
yDB	4.8
mass	2.0
xObj	-2.2
yObj	4.8
x	~~-2.2~~ -1.2
y	4.8

0x350

HealthPotion

xPotion	5.0
yPotion	-3.5
volume	6
xObj	5.0
yObj	-3.5
x	5.0
y	-3.5

0x200

in/out

2.0

```
abstract class PhysicsObject(var x: Double, var y: Double) {}
```

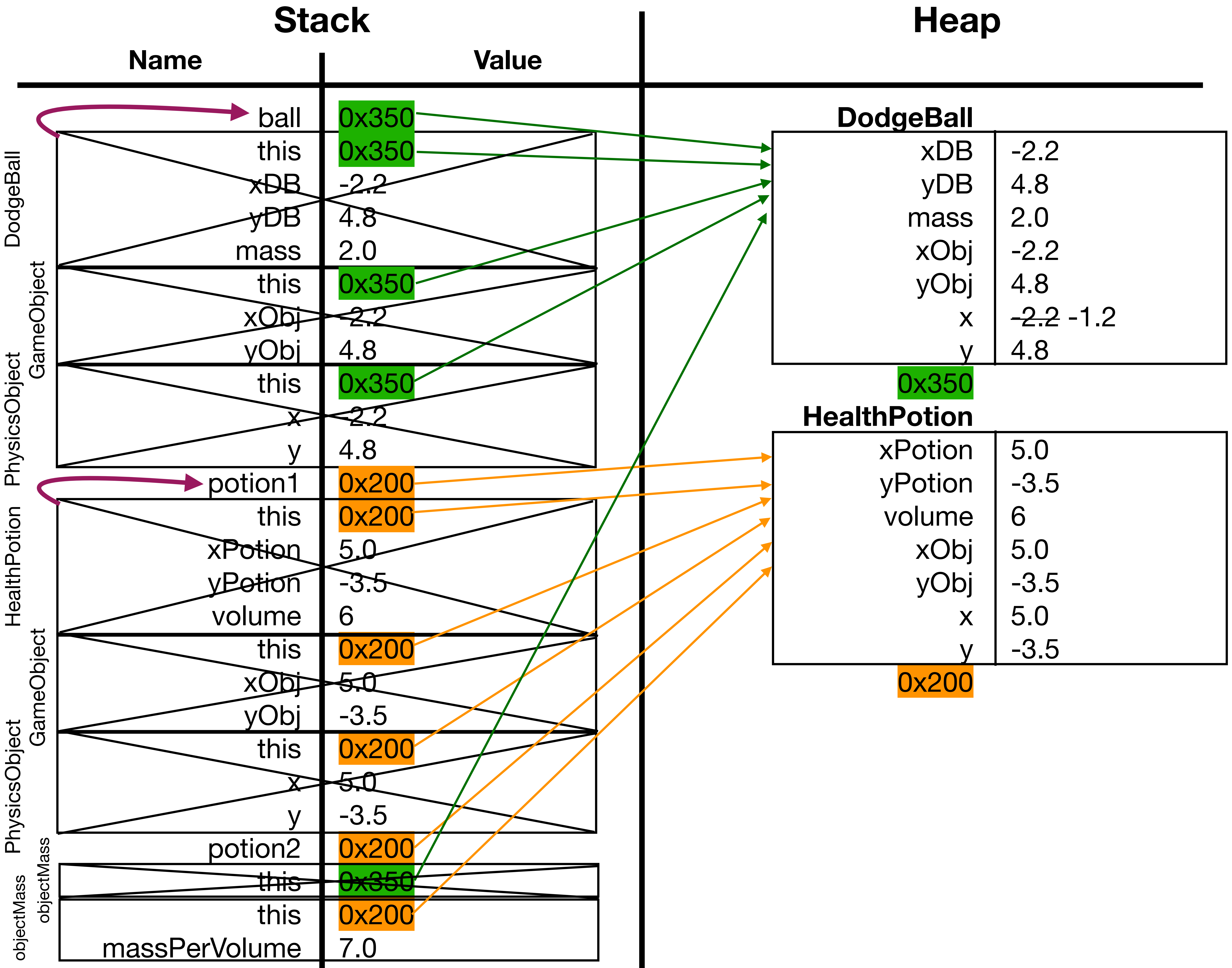
```
abstract class GameObject(var xObj: Double, var yObj: Double)
  extends PhysicsObject(xObj, yObj) {

  def objectMass(): Double

  override def toString: String = {
    "(" + this.x + ", " + this.y + "); mass: " + this.objectMass()
  }
}
```

```
class DodgeBall(var xDB: Double, var yDB: Double, val mass: Double)
  extends GameObject(xDB, yDB) {

  override def objectMass(): Double = {
    this.mass
  }
}
```

```
class HealthPotion(var xPotion: Double, var yPotion: Double,
                   val volume: Int)
  extends GameObject(xPotion, yPotion) {

  override def objectMass(): Double = {
    val massPerVolume: Double = 7.0
    this.volume * massPerVolume
  }

  override def toString: String = {
    "(" + this.x + ", " + this.y + "); volume: " + this.volume
  }
}
```

```
def main(args: Array[String]): Unit = {

  val ball: DodgeBall = new DodgeBall(-2.2, 4.8, 2)
  val potion1: HealthPotion = new HealthPotion(5.0, -3.5, 6)
  val potion2: HealthPotion = potion1

  ball.x += 1.0

  println(ball.objectMass())
  println(potion2.objectMass())
  println(ball.toString())
  println(potion1.toString())
}
```

- potion2 refers to a HealthPotion
- Use the HealthPotion objectMass method



Stack

	Name	Value
	ball	0x350
DodgeBall	this	0x350
	xDB	-2.2
	yDB	4.8
	mass	2.0
GameObject	this	0x350
	xObj	-2.2
	yObj	4.8
PhysicsObject	this	0x350
	x	-2.2
	y	4.8
	potion1	0x200
HealthPotion	this	0x200
	xPotion	5.0
	yPotion	-3.5
	volume	6
GameObject	this	0x200
	xObj	5.0
	yObj	-3.5
PhysicsObject	this	0x200
	x	5.0
	y	-3.5
	potion2	0x200
objectMass	this	0x350
objectMass	this	0x200
	massPerVolume	7.0

Heap

DodgeBall

xDB	-2.2
yDB	4.8
mass	2.0
xObj	-2.2
yObj	4.8
x	~~-2.2~~ -1.2
y	4.8

0x350

HealthPotion

xPotion	5.0
yPotion	-3.5
volume	6
xObj	5.0
yObj	-3.5
x	5.0
y	-3.5

0x200

in/out

2.0

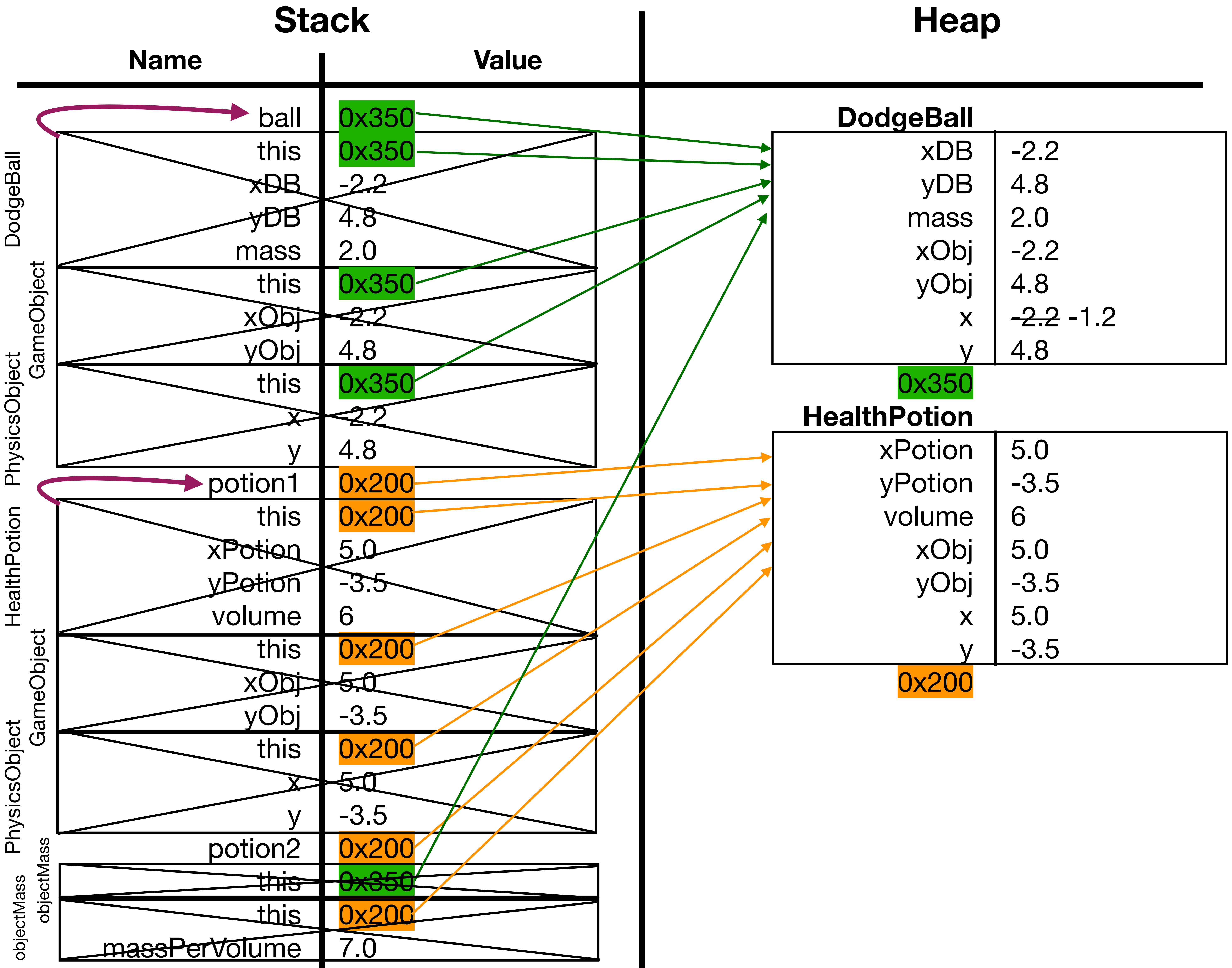
```
abstract class PhysicsObject(var x: Double, var y: Double) {}
```

```
abstract class GameObject(var xObj: Double, var yObj: Double)
  extends PhysicsObject(xObj, yObj) {

  def objectMass(): Double

  override def toString: String = {
    "(" + this.x + ", " + this.y + "); mass: " + this.objectMass()
  }
}
```

```
class DodgeBall(var xDB: Double, var yDB: Double, val mass: Double)
  extends GameObject(xDB, yDB) {

  override def objectMass(): Double = {
    this.mass
  }
}
```

```
class HealthPotion(var xPotion: Double, var yPotion: Double,
                   val volume: Int)
  extends GameObject(xPotion, yPotion) {

  override def objectMass(): Double = {
    val massPerVolume: Double = 7.0
    this.volume * massPerVolume
  }

  override def toString: String = {
    "(" + this.x + ", " + this.y + "); volume: " + this.volume
  }
}
```

```
def main(args: Array[String]): Unit = {

  val ball: DodgeBall = new DodgeBall(-2.2, 4.8, 2)
  val potion1: HealthPotion = new HealthPotion(5.0, -3.5, 6)
  val potion2: HealthPotion = potion1

  ball.x += 1.0

  println(ball.objectMass())
  println(potion2.objectMass())
  println(ball.toString())
  println(potion1.toString())
}
```



Stack

	Name	Value
	ball	0x350
DodgeBall	this	0x350
	xDB	-2.2
	yDB	4.8
	mass	2.0
GameObject	this	0x350
	xObj	-2.2
	yObj	4.8
PhysicsObject	this	0x350
	x	-2.2
	y	4.8
	potion1	0x200
HealthPotion	this	0x200
	xPotion	5.0
	yPotion	-3.5
	volume	6
GameObject	this	0x200
	xObj	5.0
	yObj	-3.5
PhysicsObject	this	0x200
	x	5.0
	y	-3.5
	potion2	0x200
objectMass	this	0x350
objectMass	this	0x200
	massPerVolume	7.0

Heap

DodgeBall (0x350)

xDB	-2.2
yDB	4.8
mass	2.0
xObj	-2.2
yObj	4.8
x	~~-2.2~~ -1.2
y	4.8

HealthPotion (0x200)

xPotion	5.0
yPotion	-3.5
volume	6
xObj	5.0
yObj	-3.5
x	5.0
y	-3.5

in/out

2.0
42.0

- Stack frame returns 42.0 to println
- Print 42.0 to the screen

```
abstract class PhysicsObject(var x: Double, var y: Double) {}
```

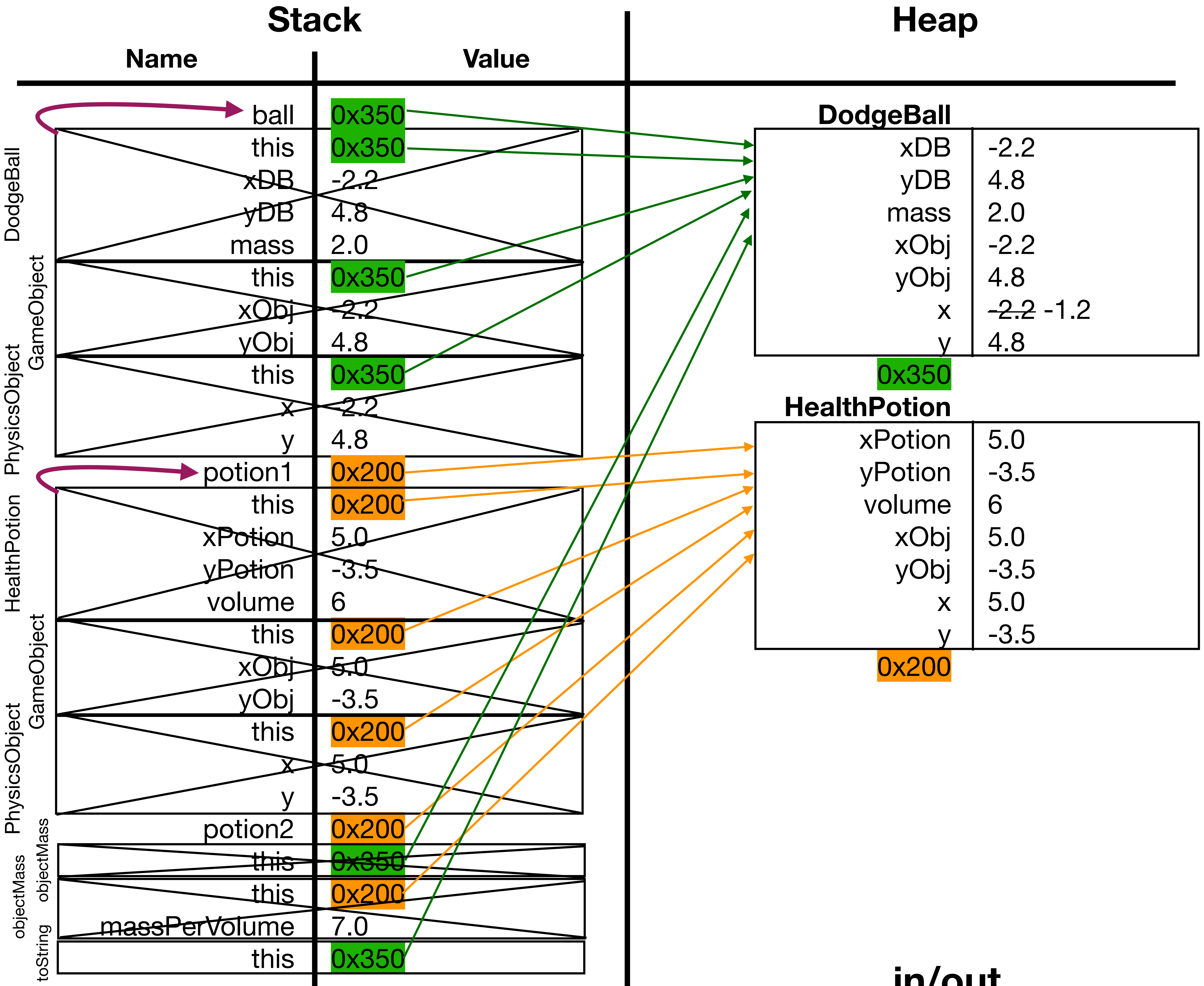
```
abstract class GameObject(var xObj: Double, var yObj: Double)
  extends PhysicsObject(xObj, yObj) {

  def objectMass(): Double

  override def toString: String = {
    "(" + this.x + ", " + this.y + "); mass: " + this.objectMass()
  }
}
```

```
class DodgeBall(var xDB: Double, var yDB: Double, val mass: Double)
  extends GameObject(xDB, yDB) {

  override def objectMass(): Double = {
    this.mass
  }
}
```

```
class HealthPotion(var xPotion: Double, var yPotion: Double,
                   val volume: Int)
  extends GameObject(xPotion, yPotion) {

  override def objectMass(): Double = {
    val massPerVolume: Double = 7.0
    this.volume * massPerVolume
  }

  override def toString: String = {
    "(" + this.x + ", " + this.y + "); volume: " + this.volume
  }
}
```

```
def main(args: Array[String]): Unit = {

  val ball: DodgeBall = new DodgeBall(-2.2, 4.8, 2)
  val potion1: HealthPotion = new HealthPotion(5.0, -3.5, 6)
  val potion2: HealthPotion = potion1

  ball.x += 1.0

  println(ball.objectMass())
  println(potion2.objectMass())
  println(ball.toString())
  println(potion1.toString())
}
```

- toString is called
- Which definition of the method is used?



Stack

	Name	Value
	ball	0x350
DodgeBall	this	0x350
	xDB	-2.2
	yDB	4.8
	mass	2.0
GameObject	this	0x350
	xObj	-2.2
	yObj	4.8
PhysicsObject	this	0x350
	x	-2.2
	y	4.8
	potion1	0x200
HealthPotion	this	0x200
	xPotion	5.0
	yPotion	-3.5
	volume	6
GameObject	this	0x200
	xObj	5.0
	yObj	-3.5
PhysicsObject	this	0x200
	x	5.0
	y	-3.5
	potion2	0x200
objectMass	this	0x350
objectMass	this	0x200
	massPerVolume	7.0
toString	this	0x350

Heap

DodgeBall

xDB	-2.2
yDB	4.8
mass	2.0
xObj	-2.2
yObj	4.8
x	~~-2.2~~ -1.2
y	4.8

0x350

HealthPotion

xPotion	5.0
yPotion	-3.5
volume	6
xObj	5.0
yObj	-3.5
x	5.0
y	-3.5

0x200

in/out

2.0
42.0

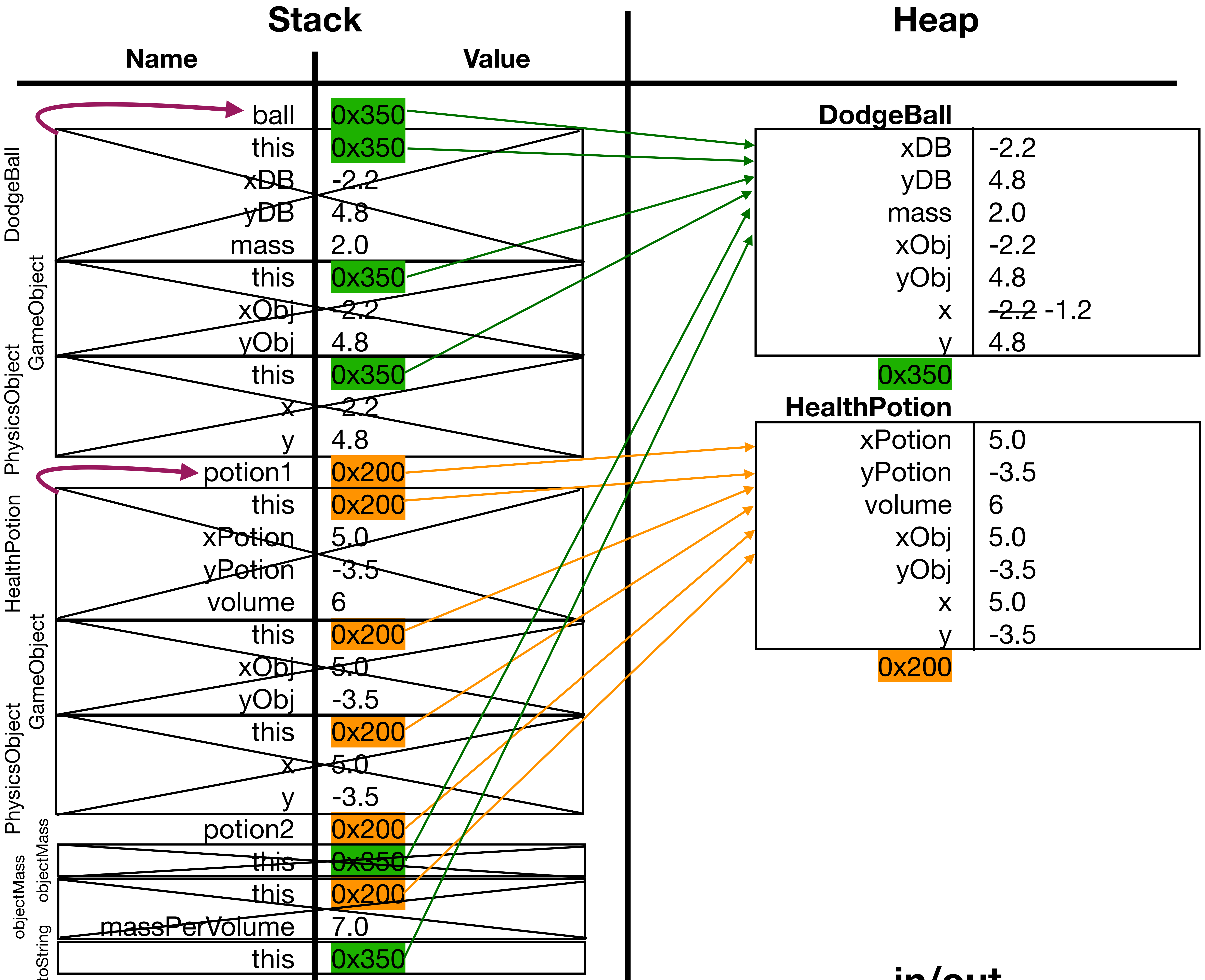
```
abstract class PhysicsObject(var x: Double, var y: Double) {}
```

```
abstract class GameObject(var xObj: Double, var yObj: Double)
  extends PhysicsObject(xObj, yObj) {

  def objectMass(): Double

  override def toString: String = {
    "(" + this.x + ", " + this.y + "); mass: " + this.objectMass()
  }
}
```

```
class DodgeBall(var xDB: Double, var yDB: Double, val mass: Double)
  extends GameObject(xDB, yDB) {

  override def objectMass(): Double = {
    this.mass
  }
}
```

```
class HealthPotion(var xPotion: Double, var yPotion: Double,
                   val volume: Int)
  extends GameObject(xPotion, yPotion) {

  override def objectMass(): Double = {
    val massPerVolume: Double = 7.0
    this.volume * massPerVolume
  }

  override def toString: String = {
    "(" + this.x + ", " + this.y + "); volume: " + this.volume
  }
}
```

```
def main(args: Array[String]): Unit = {

  val ball: DodgeBall = new DodgeBall(-2.2, 4.8, 2)
  val potion1: HealthPotion = new HealthPotion(5.0, -3.5, 6)
  val potion2: HealthPotion = potion1

  ball.x += 1.0

  println(ball.objectMass())
  println(potion2.objectMass())
  println(ball.toString())
  println(potion1.toString())
}
```



Stack

	Name	Value
	ball	0x350
DodgeBall	this	0x350
	xDB	-2.2
	yDB	4.8
	mass	2.0
GameObject	this	0x350
	xObj	-2.2
	yObj	4.8
PhysicsObject	this	0x350
	x	-2.2
	y	4.8
	potion1	0x200
HealthPotion	this	0x200
	xPotion	5.0
	yPotion	-3.5
	volume	6
GameObject	this	0x200
	xObj	5.0
	yObj	-3.5
PhysicsObject	this	0x200
	x	5.0
	y	-3.5
	potion2	0x200
objectMass	this	0x350
objectMass	this	0x200
	massPerVolume	7.0
toString	this	0x350

Heap

DodgeBall

xDB	-2.2
yDB	4.8
mass	2.0
xObj	-2.2
yObj	4.8
x	~~-2.2~~ -1.2
y	4.8

0x350

HealthPotion

xPotion	5.0
yPotion	-3.5
volume	6
xObj	5.0
yObj	-3.5
x	5.0
y	-3.5

0x200

in/out

2.0
42.0

- The reference refers to an object of type DodgeBall. Use the DodgeBall toString!
- Uh Oh. No method definition... 🤔

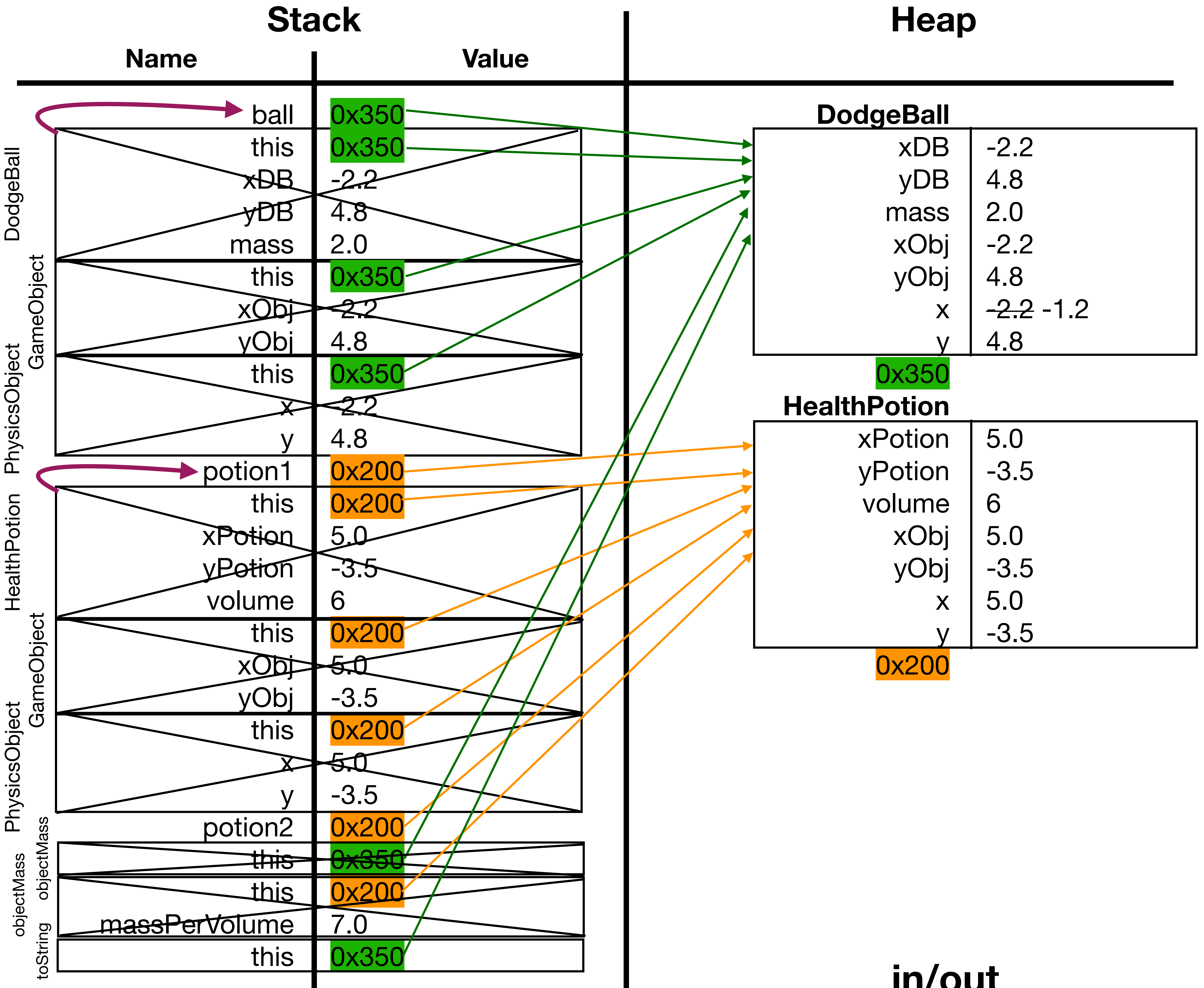
```scala
abstract class PhysicsObject(var x: Double, var y: Double) {}
```

```scala
abstract class GameObject(var xObj: Double, var yObj: Double)
  extends PhysicsObject(xObj, yObj) {

  def objectMass(): Double

  override def toString: String = {
    "(" + this.x + ", " + this.y + "); mass: " + this.objectMass()
  }
}
```

```scala
class DodgeBall(var xDB: Double, var yDB: Double, val mass: Double)
  extends GameObject(xDB, yDB) {

  override def objectMass(): Double = {
    this.mass
  }
}
```

```scala
class HealthPotion(var xPotion: Double, var yPotion: Double,
                   val volume: Int)
  extends GameObject(xPotion, yPotion) {

  override def objectMass(): Double = {
    val massPerVolume: Double = 7.0
    this.volume * massPerVolume
  }

  override def toString: String = {
    "(" + this.x + ", " + this.y + "); volume: " + this.volume
  }
}
```

```scala
def main(args: Array[String]): Unit = {

  val ball: DodgeBall = new DodgeBall(-2.2, 4.8, 2)
  val potion1: HealthPotion = new HealthPotion(5.0, -3.5, 6)
  val potion2: HealthPotion = potion1

  ball.x += 1.0

  println(ball.objectMass())
  println(potion2.objectMass())
  println(ball.toString())
  println(potion1.toString())
}
```



Stack

	Name	Value
	ball	0x350
DodgeBall	this	0x350
	xDB	-2.2
	yDB	4.8
	mass	2.0
GameObject	this	0x350
	xObj	-2.2
	yObj	4.8
PhysicsObject	this	0x350
	x	-2.2
	y	4.8
	potion1	0x200
HealthPotion	this	0x200
	xPotion	5.0
	yPotion	-3.5
	volume	6
GameObject	this	0x200
	xObj	5.0
	yObj	-3.5
PhysicsObject	this	0x200
	x	5.0
	y	-3.5
	potion2	0x200
objectMass	this	0x350
objectMass	this	0x200
	massPerVolume	7.0
toString	this	0x350

Heap

DodgeBall

xDB	-2.2
yDB	4.8
mass	2.0
xObj	-2.2
yObj	4.8
x	~~-2.2~~ -1.2
y	4.8

0x350

HealthPotion

xPotion	5.0
yPotion	-3.5
volume	6
xObj	5.0
yObj	-3.5
x	5.0
y	-3.5

0x200

in/out

```
2.0
42.0
```

- Solution: check the **super class**
- DodgeBall **inherited** toString from GameObject

```
abstract class PhysicsObject(var x: Double, var y: Double) {}
```

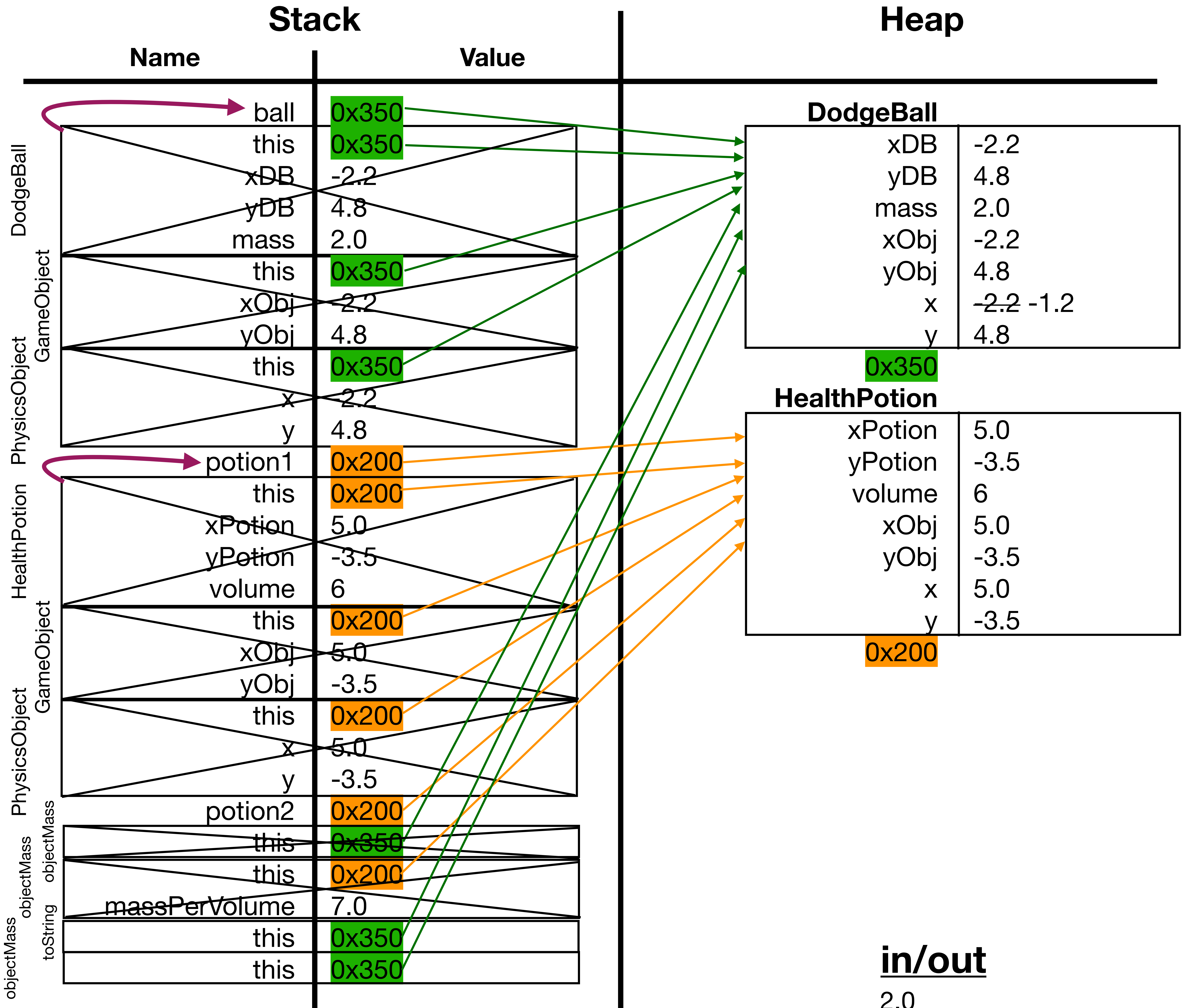
```
abstract class GameObject(var xObj: Double, var yObj: Double)
  extends PhysicsObject(xObj, yObj) {

  def objectMass(): Double

  override def toString: String = {
    "(" + this.x + ", " + this.y + "); mass: " + this.objectMass()
  }
}
```

```
class DodgeBall(var xDB: Double, var yDB: Double, val mass: Double)
  extends GameObject(xDB, yDB) {

  override def objectMass(): Double = {
    this.mass
  }
}
```

```
class HealthPotion(var xPotion: Double, var yPotion: Double,
                   val volume: Int)
  extends GameObject(xPotion, yPotion) {

  override def objectMass(): Double = {
    val massPerVolume: Double = 7.0
    this.volume * massPerVolume
  }

  override def toString: String = {
    "(" + this.x + ", " + this.y + "); volume: " + this.volume
  }
}
```

```
def main(args: Array[String]): Unit = {

  val ball: DodgeBall = new DodgeBall(-2.2, 4.8, 2)
  val potion1: HealthPotion = new HealthPotion(5.0, -3.5, 6)
  val potion2: HealthPotion = potion1

  ball.x += 1.0

  println(ball.objectMass())
  println(potion2.objectMass())
  println(ball.toString())
  println(potion1.toString())
}
```

- This toString method calls objectMass
- Since this is a reference to a DodgeBall, use DodgeBall's objectMass



Stack

	Name	Value
	ball	0x350
DodgeBall	~~this~~	~~0x350~~
	~~xDB~~	~~-2.2~~
	~~yDB~~	~~4.8~~
	~~mass~~	~~2.0~~
GameObject	~~this~~	~~0x350~~
	~~xObj~~	~~-2.2~~
	~~yObj~~	~~4.8~~
PhysicsObject	~~this~~	~~0x350~~
	~~x~~	~~-2.2~~
	~~y~~	~~4.8~~
	potion1	0x200
HealthPotion	~~this~~	~~0x200~~
	~~xPotion~~	~~5.0~~
	~~yPotion~~	~~-3.5~~
	~~volume~~	~~6~~
GameObject	~~this~~	~~0x200~~
	~~xObj~~	~~5.0~~
	~~yObj~~	~~-3.5~~
PhysicsObject	~~this~~	~~0x200~~
	~~x~~	~~5.0~~
	~~y~~	~~-3.5~~
	potion2	0x200
objectMass	~~this~~	~~0x350~~
objectMass	~~this~~	~~0x200~~
	~~massPerVolume~~	~~7.0~~
toString	this	0x350
objectMass	this	0x350

Heap

DodgeBall

xDB	-2.2
yDB	4.8
mass	2.0
xObj	-2.2
yObj	4.8
x	~~-2.2~~ -1.2
y	4.8

0x350

HealthPotion

xPotion	5.0
yPotion	-3.5
volume	6
xObj	5.0
yObj	-3.5
x	5.0
y	-3.5

0x200

in/out

2.0
42.0

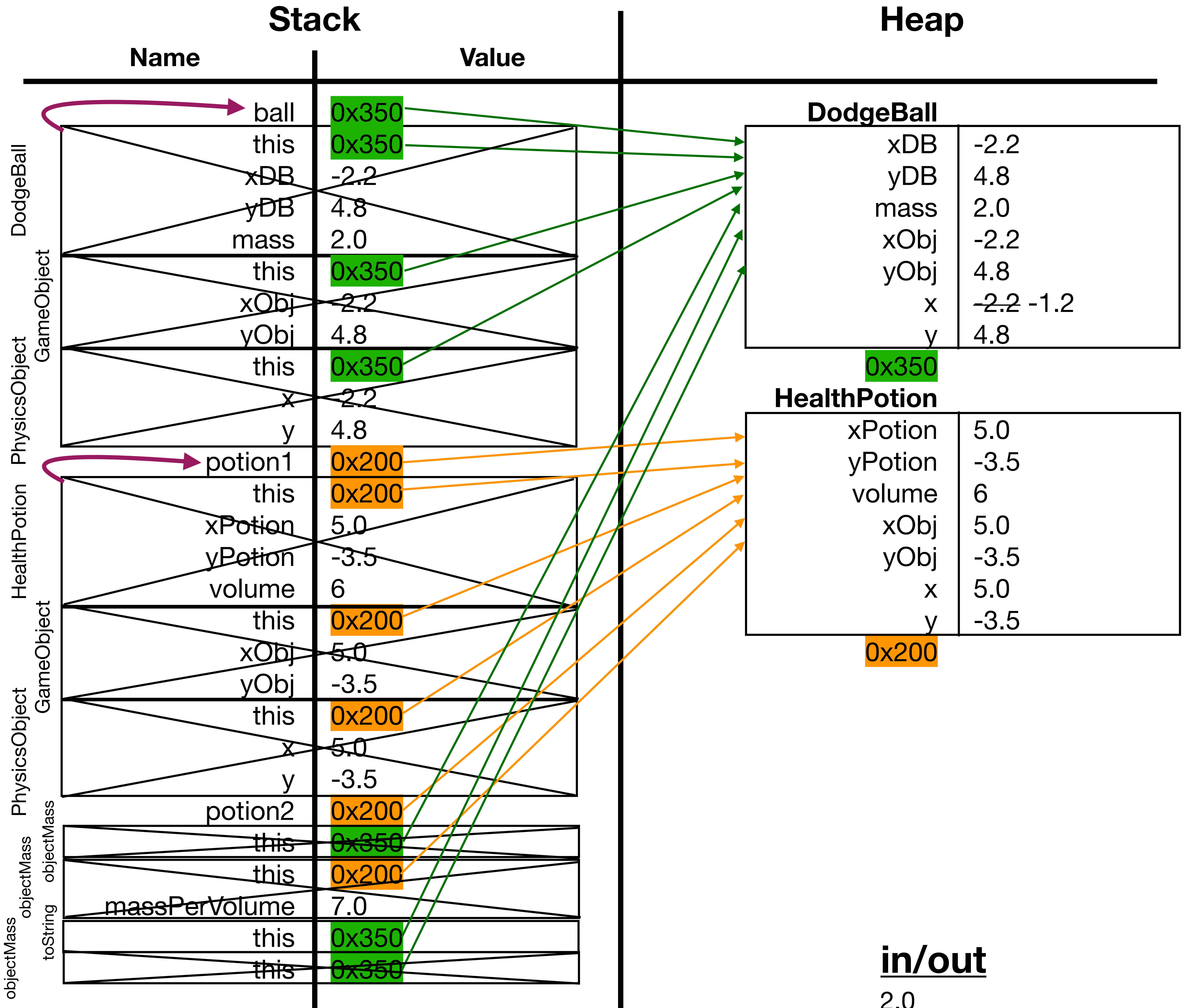
```
abstract class PhysicsObject(var x: Double, var y: Double) {}
```

```
abstract class GameObject(var xObj: Double, var yObj: Double)
  extends PhysicsObject(xObj, yObj) {

  def objectMass(): Double

  override def toString: String = {
    "(" + this.x + ", " + this.y + "); mass: " + this.objectMass()
  }
}
```

```
class DodgeBall(var xDB: Double, var yDB: Double, val mass: Double)
  extends GameObject(xDB, yDB) {

  override def objectMass(): Double = {
    this.mass
  }
}
```

```
class HealthPotion(var xPotion: Double, var yPotion: Double,
                   val volume: Int)
  extends GameObject(xPotion, yPotion) {

  override def objectMass(): Double = {
    val massPerVolume: Double = 7.0
    this.volume * massPerVolume
  }

  override def toString: String = {
    "(" + this.x + ", " + this.y + "); volume: " + this.volume
  }
}
```

```
def main(args: Array[String]): Unit = {

  val ball: DodgeBall = new DodgeBall(-2.2, 4.8, 2)
  val potion1: HealthPotion = new HealthPotion(5.0, -3.5, 6)
  val potion2: HealthPotion = potion1

  ball.x += 1.0

  println(ball.objectMass())
  println(potion2.objectMass())
  println(ball.toString())
  println(potion1.toString())
}
```



Stack

Name	Value
ball	0x350
this	0x350
xDB	-2.2
yDB	4.8
mass	2.0
this	0x350
xObj	-2.2
yObj	4.8
this	0x350
x	-2.2
y	4.8
potion1	0x200
this	0x200
xPotion	5.0
yPotion	-3.5
volume	6
this	0x200
xObj	5.0
yObj	-3.5
this	0x200
x	5.0
y	-3.5
potion2	0x200
this	0x350
this	0x200
massPerVolume	7.0
this	0x350
this	0x350

Heap

DodgeBall

xDB	-2.2
yDB	4.8
mass	2.0
xObj	-2.2
yObj	4.8
x	~~-2.2~~ -1.2
y	4.8

0x350

HealthPotion

xPotion	5.0
yPotion	-3.5
volume	6
xObj	5.0
yObj	-3.5
x	5.0
y	-3.5

0x200

in/out

2.0
42.0

- objectMass returns 2.0
- Concatenate the Strings and return

```
abstract class PhysicsObject(var x: Double, var y: Double) {}
```

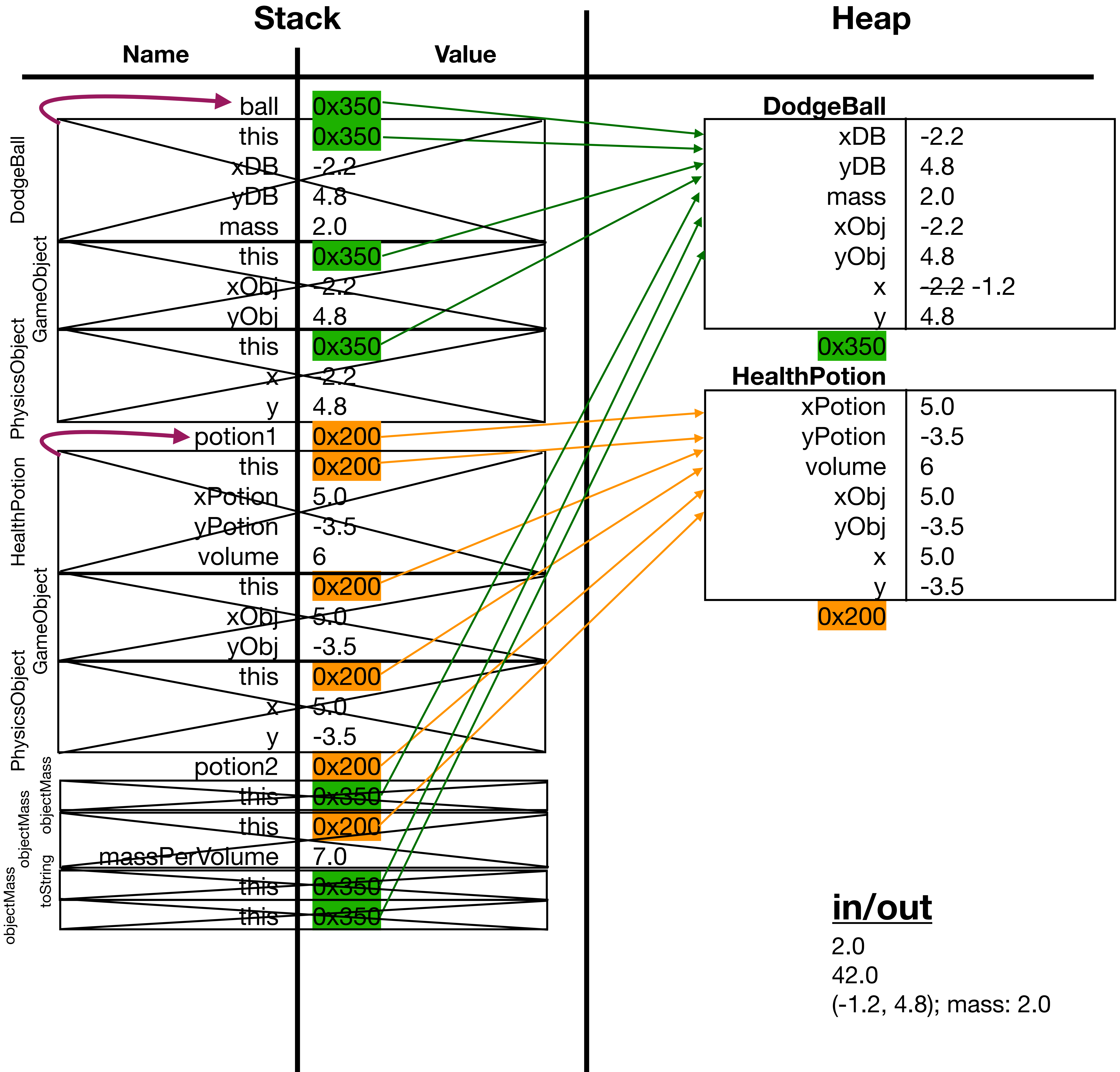
```
abstract class GameObject(var xObj: Double, var yObj: Double)
  extends PhysicsObject(xObj, yObj) {

  def objectMass(): Double

  override def toString: String = {
    "(" + this.x + ", " + this.y + "); mass: " + this.objectMass()
  }
}
```

```
class DodgeBall(var xDB: Double, var yDB: Double, val mass: Double)
  extends GameObject(xDB, yDB) {

  override def objectMass(): Double = {
    this.mass
  }
}
```

```
class HealthPotion(var xPotion: Double, var yPotion: Double,
                   val volume: Int)
  extends GameObject(xPotion, yPotion) {

  override def objectMass(): Double = {
    val massPerVolume: Double = 7.0
    this.volume * massPerVolume
  }

  override def toString: String = {
    "(" + this.x + ", " + this.y + "); volume: " + this.volume
  }
}
```

```
def main(args: Array[String]): Unit = {

  val ball: DodgeBall = new DodgeBall(-2.2, 4.8, 2)
  val potion1: HealthPotion = new HealthPotion(5.0, -3.5, 6)
  val potion2: HealthPotion = potion1

  ball.x += 1.0

  println(ball.objectMass())
  println(potion2.objectMass())
  println(ball.toString())
  println(potion1.toString())
}
```

- Print to the screen



Stack

	Name	Value
	ball	0x350
DodgeBall	this	0x350
	xDB	-2.2
	yDB	4.8
	mass	2.0
GameObject	this	0x350
	xObj	-2.2
	yObj	4.8
PhysicsObject	this	0x350
	x	-2.2
	y	4.8
	potion1	0x200
HealthPotion	this	0x200
	xPotion	5.0
	yPotion	-3.5
	volume	6
GameObject	this	0x200
	xObj	5.0
	yObj	-3.5
PhysicsObject	this	0x200
	x	5.0
	y	-3.5
	potion2	0x200
objectMass	this	0x350
objectMass	this	0x200
	massPerVolume	7.0
toString	this	0x350
objectMass	this	0x350

Heap

DodgeBall

xDB	-2.2
yDB	4.8
mass	2.0
xObj	-2.2
yObj	4.8
x	~~-2.2~~ -1.2
y	4.8

0x350

HealthPotion

xPotion	5.0
yPotion	-3.5
volume	6
xObj	5.0
yObj	-3.5
x	5.0
y	-3.5

0x200

in/out

2.0
42.0
(-1.2, 4.8); mass: 2.0

```
abstract class PhysicsObject(var x: Double, var y: Double) {}
```

```
abstract class GameObject(var xObj: Double, var yObj: Double)
  extends PhysicsObject(xObj, yObj) {

  def objectMass(): Double

  override def toString: String = {
    "(" + this.x + ", " + this.y + "); mass: " + this.objectMass()
  }
}
```

```
class DodgeBall(var xDB: Double, var yDB: Double, val mass: Double)
  extends GameObject(xDB, yDB) {

  override def objectMass(): Double = {
    this.mass
  }
}
```

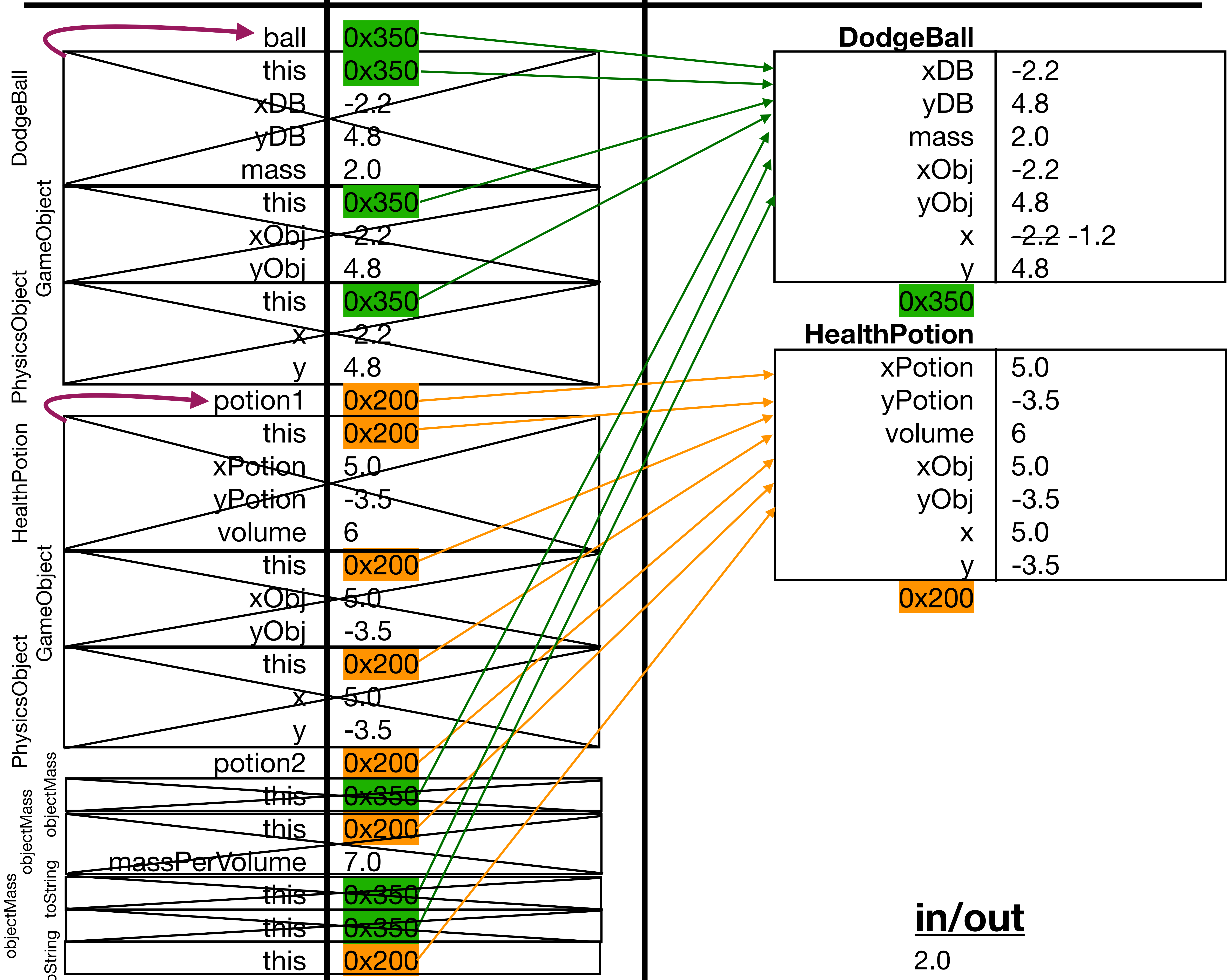
```
class HealthPotion(var xPotion: Double, var yPotion: Double,
                   val volume: Int)
  extends GameObject(xPotion, yPotion) {

  override def objectMass(): Double = {
    val massPerVolume: Double = 7.0
    this.volume * massPerVolume
  }

  override def toString: String = {
    "(" + this.x + ", " + this.y + "); volume: " + this.volume
  }
}
```

```
def main(args: Array[String]): Unit = {

  val ball: DodgeBall = new DodgeBall(-2.2, 4.8, 2)
  val potion1: HealthPotion = new HealthPotion(5.0, -3.5, 6)
  val potion2: HealthPotion = potion1

  ball.x += 1.0

  println(ball.objectMass())
  println(potion2.objectMass())
  println(ball.toString())
  println(potion1.toString())
}
```

- potion1 refers to a HealthPotion
- HealthPotion overrides the GameObject toString

Stack

Heap



	Name	Value
	ball	0x350
DodgeBall	this	0x350
	xDB	-2.2
	yDB	4.8
	mass	2.0
GameObject	this	0x350
	xObj	-2.2
	yObj	4.8
PhysicsObject	this	0x350
	x	-2.2
	y	4.8
	potion1	0x200
HealthPotion	this	0x200
	xPotion	5.0
	yPotion	-3.5
	volume	6
GameObject	this	0x200
	xObj	5.0
	yObj	-3.5
PhysicsObject	this	0x200
	x	5.0
	y	-3.5
	potion2	0x200
objectMass	this	0x350
objectMass	this	0x200
	massPerVolume	7.0
toString	this	0x350
objectMass	this	0x350
toString	this	0x200

DodgeBall

xDB	-2.2
yDB	4.8
mass	2.0
xObj	-2.2
yObj	4.8
x	~~-2.2~~ -1.2
y	4.8

0x350

HealthPotion

xPotion	5.0
yPotion	-3.5
volume	6
xObj	5.0
yObj	-3.5
x	5.0
y	-3.5

0x200

in/out

2.0
42.0
(-1.2, 4.8); mass: 2.0

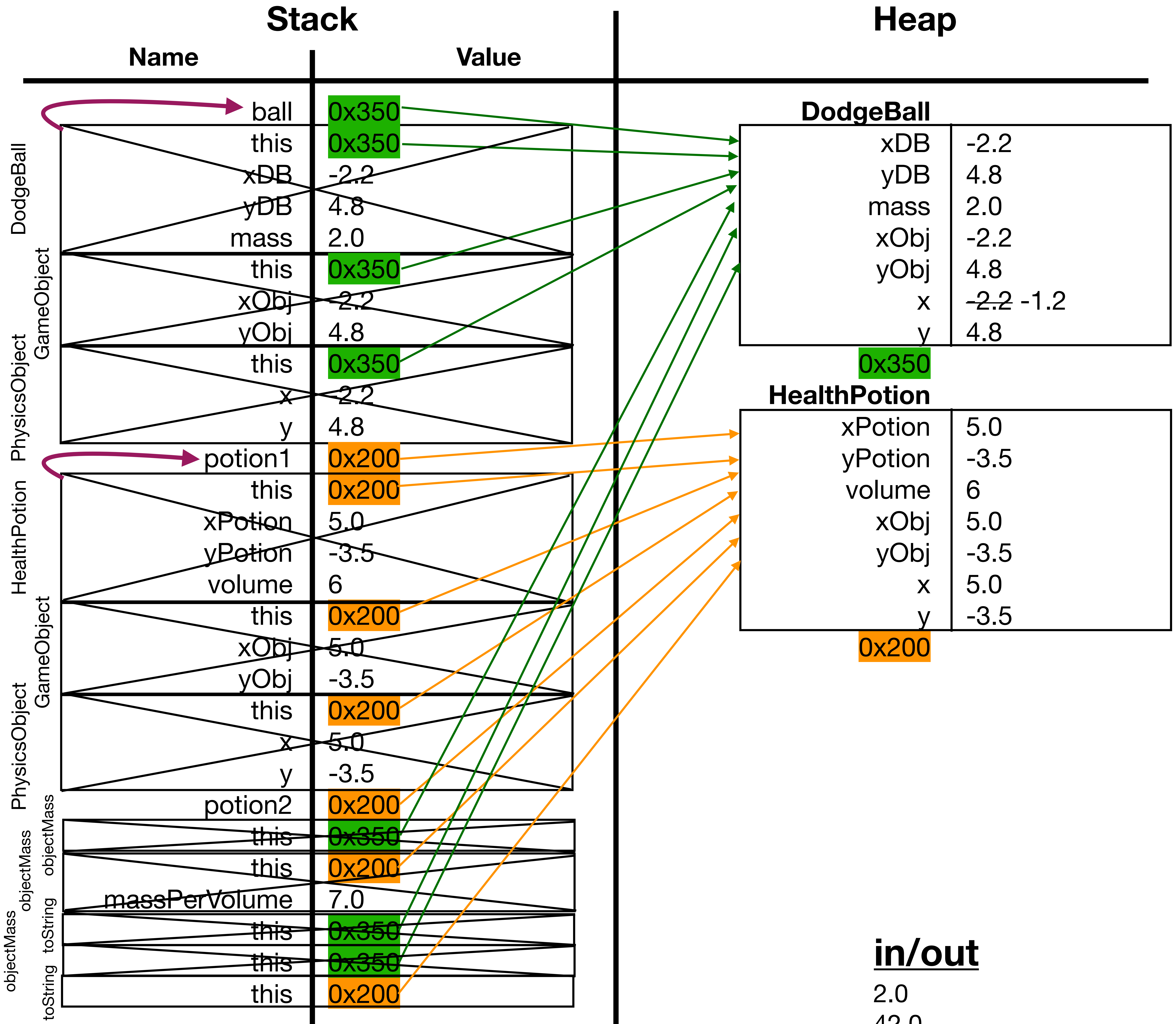
```
abstract class PhysicsObject(var x: Double, var y: Double) {}
```

```
abstract class GameObject(var xObj: Double, var yObj: Double)
  extends PhysicsObject(xObj, yObj) {

  def objectMass(): Double

  override def toString: String = {
    "(" + this.x + ", " + this.y + "); mass: " + this.objectMass()
  }
}
```

```
class DodgeBall(var xDB: Double, var yDB: Double, val mass: Double)
  extends GameObject(xDB, yDB) {

  override def objectMass(): Double = {
    this.mass
  }
}
```

```
class HealthPotion(var xPotion: Double, var yPotion: Double,
                   val volume: Int)
  extends GameObject(xPotion, yPotion) {

  override def objectMass(): Double = {
    val massPerVolume: Double = 7.0
    this.volume * massPerVolume
  }

  override def toString: String = {
    "(" + this.x + ", " + this.y + "); volume: " + this.volume
  }
}
```

```
def main(args: Array[String]): Unit = {

  val ball: DodgeBall = new DodgeBall(-2.2, 4.8, 2)
  val potion1: HealthPotion = new HealthPotion(5.0, -3.5, 6)
  val potion2: HealthPotion = potion1

  ball.x += 1.0

  println(ball.objectMass())
  println(potion2.objectMass())
  println(ball.toString())
  println(potion1.toString())
}
```



Stack

	Name	Value
	ball	0x350
DodgeBall	~~this~~	~~0x350~~
	~~xDB~~	~~-2.2~~
	~~yDB~~	~~4.8~~
	~~mass~~	~~2.0~~
GameObject	~~this~~	~~0x350~~
	~~xObj~~	~~-2.2~~
	~~yObj~~	~~4.8~~
PhysicsObject	~~this~~	~~0x350~~
	~~x~~	~~-2.2~~
	~~y~~	~~4.8~~
	potion1	0x200
HealthPotion	~~this~~	~~0x200~~
	~~xPotion~~	~~5.0~~
	~~yPotion~~	~~-3.5~~
	~~volume~~	~~6~~
GameObject	~~this~~	~~0x200~~
	~~xObj~~	~~5.0~~
	~~yObj~~	~~-3.5~~
PhysicsObject	~~this~~	~~0x200~~
	~~x~~	~~5.0~~
	~~y~~	~~-3.5~~
	potion2	0x200
objectMass	~~this~~	~~0x350~~
objectMass	~~this~~	~~0x200~~
	~~massPerVolume~~	~~7.0~~
toString	~~this~~	~~0x350~~
objectMass	~~this~~	~~0x350~~
toString	this	0x200

Heap

DodgeBall

xDB	-2.2
yDB	4.8
mass	2.0
xObj	-2.2
yObj	4.8
x	~~-2.2~~ -1.2
y	4.8

0x350

HealthPotion

xPotion	5.0
yPotion	-3.5
volume	6
xObj	5.0
yObj	-3.5
x	5.0
y	-3.5

0x200

in/out

```
2.0
42.0
(-1.2, 4.8); mass: 2.0
```

- Concatenate the Strings and return
- The overridden GameObject toString is never used for a HealthPotion

```
abstract class PhysicsObject(var x: Double, var y: Double) {}
```

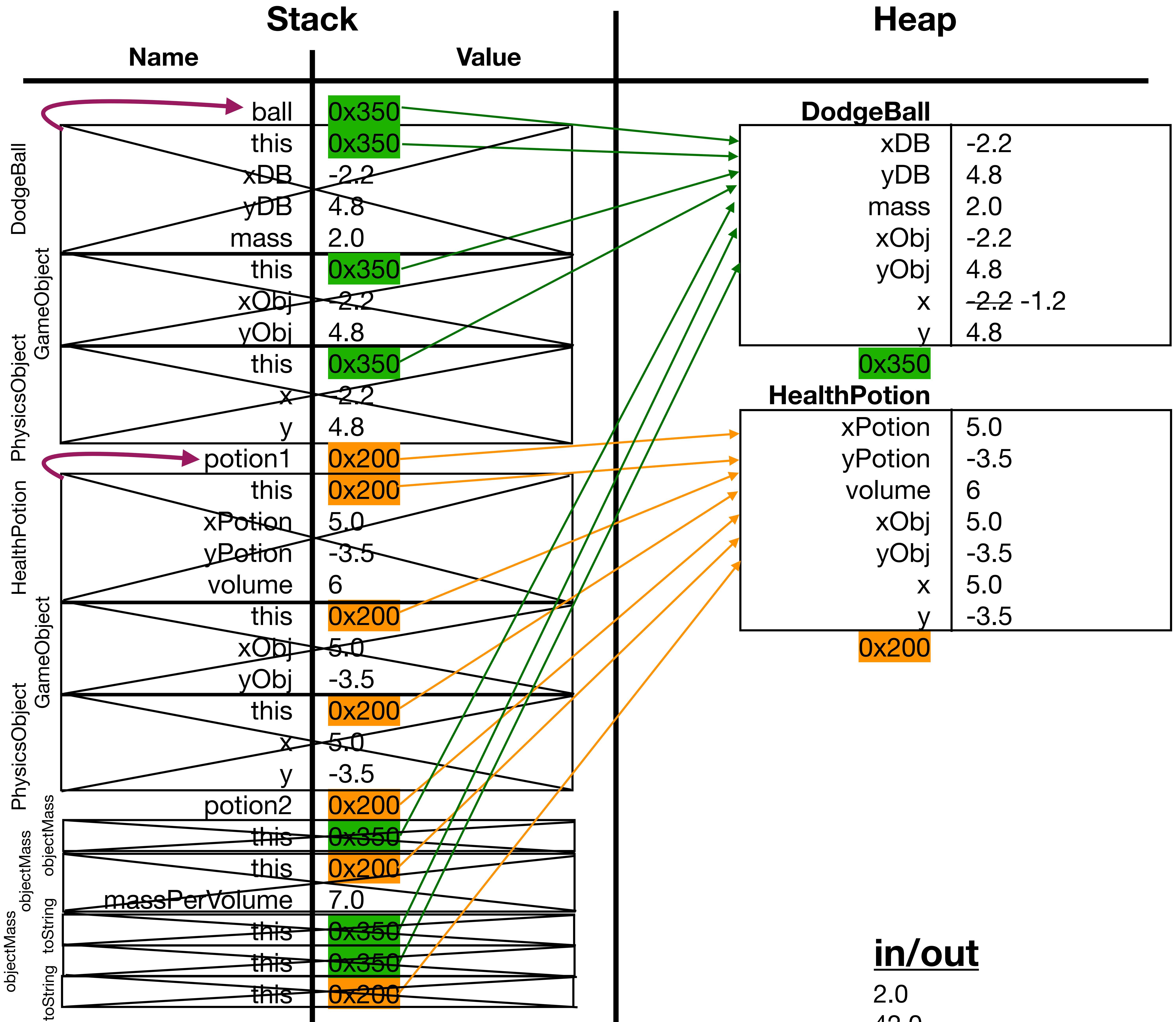
```
abstract class GameObject(var xObj: Double, var yObj: Double)
  extends PhysicsObject(xObj, yObj) {

  def objectMass(): Double

  override def toString: String = {
    "(" + this.x + ", " + this.y + "); mass: " + this.objectMass()
  }
}
```

```
class DodgeBall(var xDB: Double, var yDB: Double, val mass: Double)
  extends GameObject(xDB, yDB) {

  override def objectMass(): Double = {
    this.mass
  }
}
```

```
class HealthPotion(var xPotion: Double, var yPotion: Double,
                   val volume: Int)
  extends GameObject(xPotion, yPotion) {

  override def objectMass(): Double = {
    val massPerVolume: Double = 7.0
    this.volume * massPerVolume
  }

  override def toString: String = {
    "(" + this.x + ", " + this.y + "); volume: " + this.volume
  }
}
```

```
def main(args: Array[String]): Unit = {

  val ball: DodgeBall = new DodgeBall(-2.2, 4.8, 2)
  val potion1: HealthPotion = new HealthPotion(5.0, -3.5, 6)
  val potion2: HealthPotion = potion1

  ball.x += 1.0

  println(ball.objectMass())
  println(potion2.objectMass())
  println(ball.toString())
  println(potion1.toString())
}
```

- Print to the screen
- End the program
- End the lecture



Stack

	Name	Value
	ball	0x350
DodgeBall	this	0x350
	xDB	-2.2
	yDB	4.8
	mass	2.0
GameObject	this	0x350
	xObj	-2.2
	yObj	4.8
PhysicsObject	this	0x350
	x	-2.2
	y	4.8
	potion1	0x200
HealthPotion	this	0x200
	xPotion	5.0
	yPotion	-3.5
	volume	6
GameObject	this	0x200
	xObj	5.0
	yObj	-3.5
PhysicsObject	this	0x200
	x	5.0
	y	-3.5
	potion2	0x200
objectMass	this	0x350
objectMass	this	0x200
	massPerVolume	7.0
toString	this	0x350
objectMass	this	0x350
toString	this	0x200

Heap

DodgeBall

xDB	-2.2
yDB	4.8
mass	2.0
xObj	-2.2
yObj	4.8
x	~~-2.2~~ -1.2
y	4.8

0x350

HealthPotion

xPotion	5.0
yPotion	-3.5
volume	6
xObj	5.0
yObj	-3.5
x	5.0
y	-3.5

0x200

in/out

```
2.0
42.0
(-1.2, 4.8); mass: 2.0
(5.0, -3.5); volume: 6
```