The Gumball Machine!

Mighty Gumball Inc. wants to modernize their gumball machines. They are asking you to program the controller software which will sit between the user interface elements, and the provided hardware device.

The user interface elements consist of:

- A coin slot that can sense when a **quarter** was **inserted**.
- A button the customer uses to **request** the machine to **eject** their quarter.
- A **crank** which when **turned** releases a gumball (but only if they've paid first).
- A **reset** button which when pressed indicates the machine has been **refilled**.

The hardware device interface is your controller software's interface to the hardware consisting of:

- A **display message** operation that shows text messages on the LCD Display.
- A **dispense quarter** operation that will eject a quarter to the customer.
- A **dispense gumball** operation which releases a gumball. The hardware can detect if a gumball was released, and returns **true** when it did. If the hardware could not dispense a gumball, it means the machine has run out of gumballs.

The gumball machine has electronics added, but the hardware is not able to count the number of gumballs in the machine at any time. The success or failure of the dispense operation is the only way to determine if the device can still sell gumballs.

When the machine is powered on, it assumes it's out of gumballs until the refill button is pressed. Once refill is pressed, it will be ready to sell gumballs.

Later in this document are tables describing inputs and expected outputs.

Sample code and unit tests may be provided. Examine the existing code (if any) and complete the code so that your controller software could be used to sell gumballs in a real electronic gumball machine.

Note: the test code should only use the public methods. **Do not change** the public method signatures of either the **GumballMachine** or the **GumballHardwareDevice** interface. You may add fields or add additional classes.

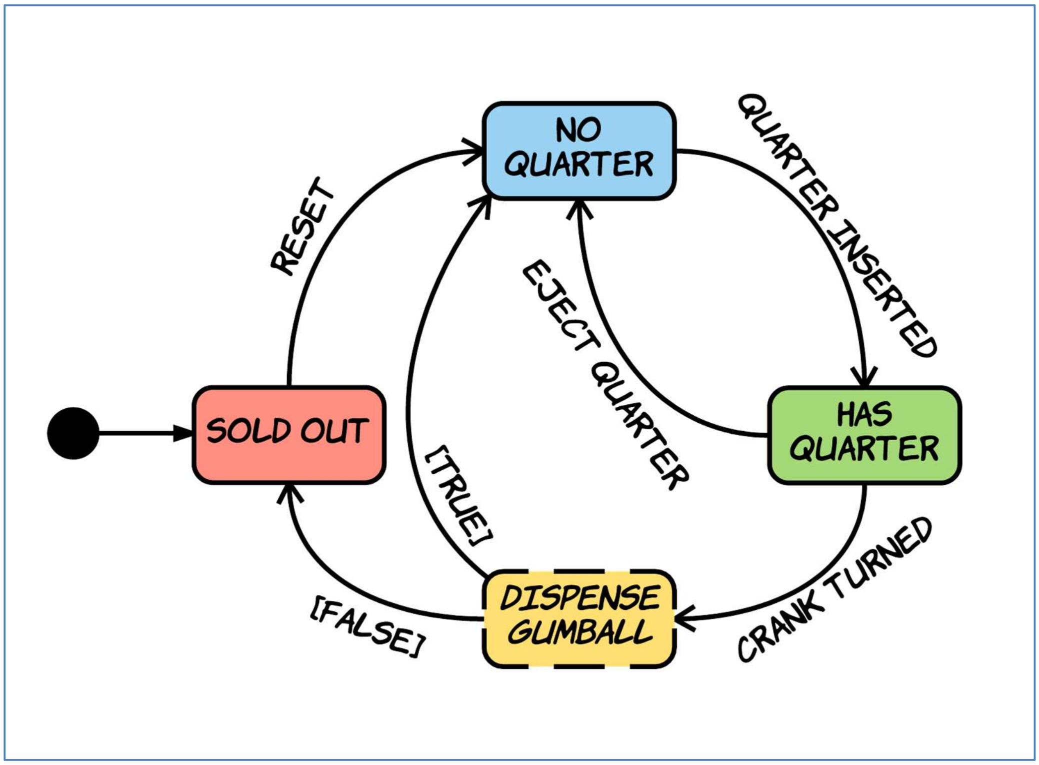


Figure 1: Gumball Machine UML State Diagram

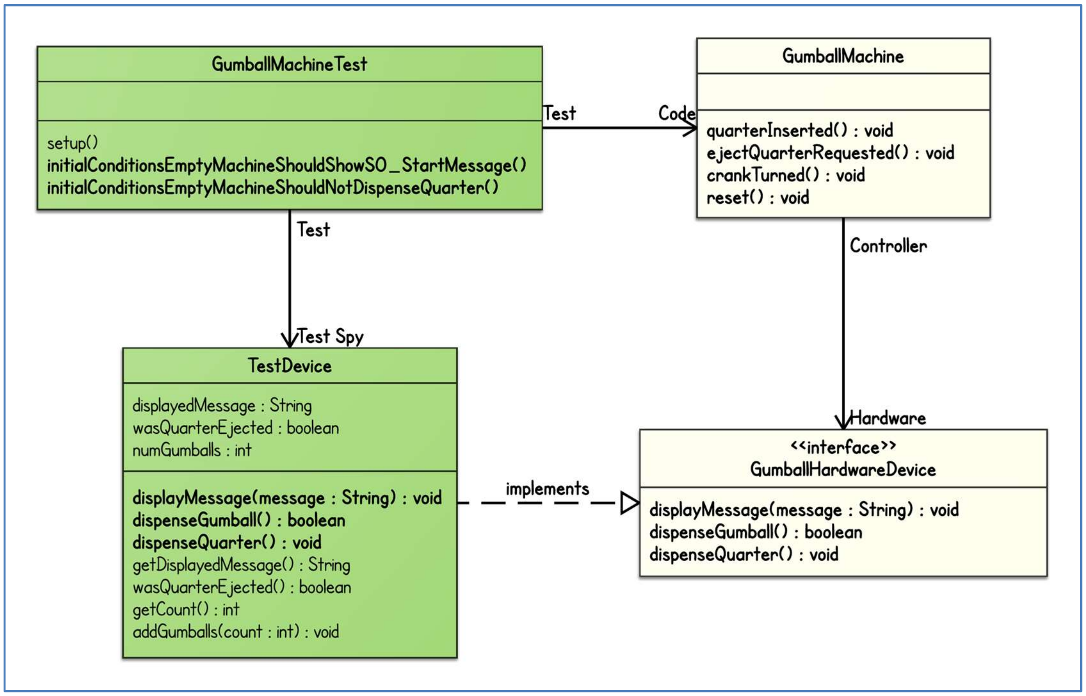
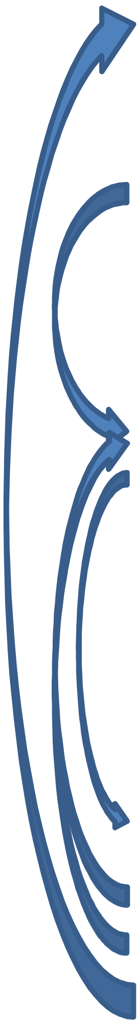

GumballMachineTest
setup()
initialConditionsEmptyMachineShouldShowSO_StartMessage()
initialConditionsEmptyMachineShouldNotDispenseQuarter()
Test
Code
GumballMachine
quarterInserted() : void
ejectQuarterRequested() : void
crankTurned() : void
reset() : void
Test
Controller
Test Spy
TestDevice
displayedMessage : String
wasQuarterEjected : boolean
numGumballs : int
displayMessage(message : String) : void
dispenseGumball() : boolean
dispenseQuarter() : void
getDisplayedMessage() : String
wasQuarterEjected() : boolean
getCount() : int
addGumballs(count : int) : void
implements
Hardware
<<interface>>
GumballHardwareDevice
displayMessage(message : String) : void
dispenseGumball() : boolean
dispenseQuarter() : void

Figure 2: Gumball Machine UML Class Diagram

Gumball Machine Messages



Gumball Machine Sold Out

User Action	Display Message?	Hardware Action?
<at Start>	Sorry, the machine is sold out.	
Quarter Inserted	There are no Gumballs, please pick up your Quarter.	dispenseQuarter()
Eject Quarter	This is not a Slot Machine.	
Crank Turned	There are no Gumballs. :(	
Reset	Quarter for a Gumball!	

Gumball Machine No Quarter

User Action	Display Message?	Hardware Action?
<at Start>	Quarter for a Gumball!	
Quarter Inserted	Turn the Crank for a Gumball!	
Eject Quarter	You haven't inserted a Quarter yet.	
Crank Turned	You need to pay first.	

Gumball Machine Has Quarter

User Action	Has Gumballs	Display Message?	Hardware Action?
<at Start>	-	Turn the Crank for a Gumball!	
Quarter Inserted	-	You can't insert another Quarter.	dispenseQuarter()
Eject Quarter	-	Pick up your Quarter from the tray.	dispenseQuarter()
Crank Turned	True	Quarter for a Gumball!	dispenseGumball()
Crank Turned	False	Sorry, the machine is sold out.	dispenseQuarter()