Toy Project  
  
Objective   
-To practice principles learned up to this point in the intersession term

Toy Class  
This class is responsible for modelling an instance of a toy object.

Properties  
+Model – read-only string property used to hold the model name of the toy.  
+IsOff – Boolean property used to modify and report the power state of the toy.

Methods  
+Toy (model : string, isOff : bool)  
 -exceptions!!! Throw exceptions if the model name is blank, and if the state of the object’s power is intitially set to “on”

+TogglePower() : void  
-used to toggle the power of the toy object.

# RcCar Class

This class is derived from the Toy Class to create an instance of an RcCar object.

## Member Variables

-battery :Battery – holds the battery object   
-speed : Speed – enumeration to set the speed state   
-hasAI: bool – Boolean value used to set the option argument of the RcCar constructor.

## Properties

+Color – String property used to modify and report the car’s color  
set exception – throw an exception if the color value is blank with an appropriate message.  
.

## Methods

+RcCar(model : string, isOff : bool, color : string, speed : Speed, hasAI : bool = false)  
-used to create an instance of an RcCar object.  
 --exceptions, throw an exceptions if the RcCar’s color value is blank, with an appropriate message.

+AddBattery(batteryType : BatteryType) : Battery  
-when invoked this method will return an instance of a battery object depending on which battery type it is passed.

+Accelerate(speed : Speed) : void  
when invoked this method will decrease the life of the battery object depending on which speed it is travelling.

# Battery Class

This abstract class is used to model the instance of a Battery object.

## Properties

+BatteryType – Enumeration used to report the battery type. <<read only>>  
+CurrentLife : integer abstract property used to modify and report the battery’s current life.

Methods  
+Battery(battType : BatteryType) – used to construct the model instance of a battery object.  
+ChargeBattery() : void - abstract method used to charge the battery.

BattCopper Class  
This class derives from the Battery class to create an instance of a BattCopper object

Member Variables  
+ MAXLIFE\_COPPER – int value used to set the maximum life = 100.  
-currentLife :int – valued used to hold the amount of life the battery currently has, initial value reflects the max life of the battery.  
-maxLife : int – value used to hold the maximum amount of life the battery can hold.

## Properties

+MaxLife : int property used to report the amount of life the battery can hold  
+CurrentLife : int property used to modify and report the current amount of battery life.  
be sure to set that if the current life of the battery reaches zero, that the power shuts off.  
!!!Exceptions!!! – throw an exception with appropriate message if the current life is greater than the maximum life, and if the current life is less than zero.  
+BattCopper(battType : BatteryType, currentLife : int, maxLife : int) – used to create an instance of a BattCopper object.  
!!!Exceptions!!! throw exceptions IF –  
🡪 Current life is less than zero  
🡪current life is greater than the max life.  
🡪max life is not equal to 100.  
+ChargeBattery() : void – when invoked this method will charge this battery by 20.

BattNiCa Class  
This class derives from the Battery class to create an instance of a BattNiCa object

Member Variables  
+ MAXLIFE\_NICA – int value used to set the maximum life = 200.  
-currentLife :int – valued used to hold the amount of life the battery currently has, initial value reflects the max life of the battery.  
-maxLife : int – value used to hold the maximum amount of life the battery can hold.

## Properties

+MaxLife : int property used to report the amount of life the battery can hold  
+CurrentLife : int property used to modify and report the current amount of battery life.  
!!!Exceptions!!! – throw an exception with appropriate message if the current life is greater than the maximum life, and if the current life is less than zero.  
+BattNiCa(battType : BatteryType, currentLife : int, maxLife : int) – used to create an instance of a BattNiCa object.  
!!!Exceptions!!! throw exceptions IF –  
🡪 Current life is less than zero  
🡪current life is greater than the max life.  
🡪max life is not equal to 200.  
+ChargeBattery() : void – when invoked this method will charge this battery by 30.

BattLi Class  
This class derives from the Battery class to create an instance of a BattCopper object

Member Variables  
+ MAXLIFE\_Li – int value used to set the maximum life = 300.  
-currentLife :int – valued used to hold the amount of life the battery currently has, initial value reflects the max life of the battery.  
-maxLife : int – value used to hold the maximum amount of life the battery can hold.

## Properties

+MaxLife : int property used to report the amount of life the battery can hold  
+CurrentLife : int property used to modify and report the current amount of battery life.  
!!!Exceptions!!! – throw an exception with appropriate message if the current life is greater than the maximum life, and if the current life is less than zero.  
+BattLi(battType : BatteryType, currentLife : int, maxLife : int) – used to create an instance of a BattLi object.  
!!!Exceptions!!! throw exceptions IF –  
🡪 Current life is less than zero  
🡪current life is greater than the max life.  
🡪max life is not equal to 300.  
+ChargeBattery() : void – when invoked this method will charge this battery by 50.

SpeedConsumption Class  
This static class is used to hold integer values for different speeds being travelled.

Static Variables  
+NONE : int = 0  
+SLOW : int =-5  
+MEDIUM : int = -10  
+Fast : int = -15  
+HYPER\_SPEED = -30.

Speed Class  
This Enumeration class defines different speeds the object can travel at

## Variables

None,Slow, Medium, Fast, HyperSpeed

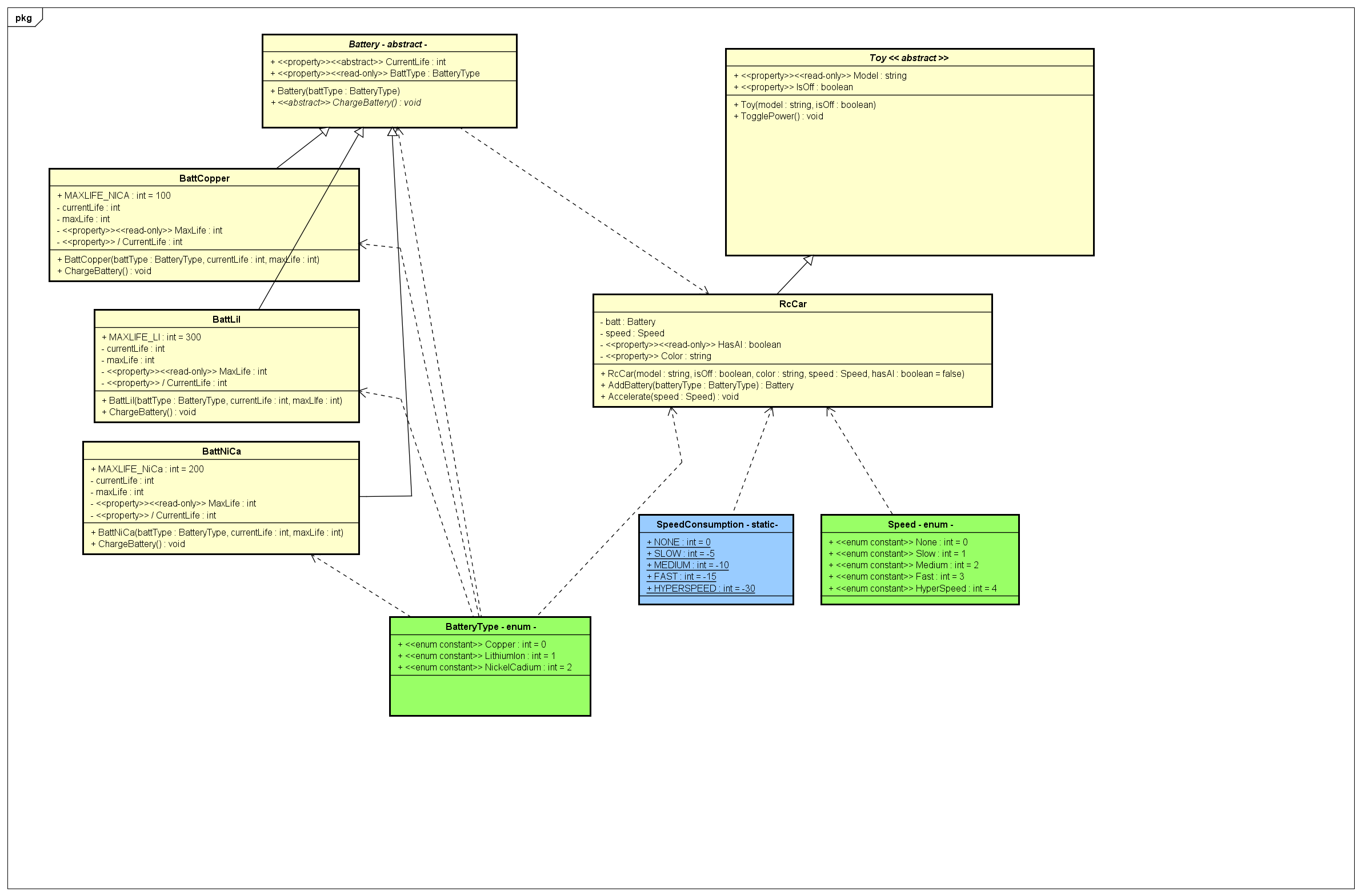
BatteryType Class  
This enumeration class defines different types of batteries

## Variables

Copper, NickelCadium, LithiumIon

Functionality  
code the above classes to meet the requirements I set out for you, based also on the UML diagrams I provided.   
- I suggest you Unit test for every outcome, and write a test program to ensure proper functionality –

Have Fun!



## 