|  |
| --- |
| **Program 01** |
| **Output** |
| >>>  RESTART: /Users/biniamlemma/Desktop/CSCI\_2061/Assn\_08/Assn 08 Pr 01 Start.py  Standard time:  (8, 36, 0, 'PM')  8:36:0 PM  Standard date:  (13, 10, 2015)  October,13 2015  Date time:  (8, 36, 0, 'PM', 13, 10, 2015)  8:36:0 PM  October,13 2015  Military time:  (2236, 22)  2236 22  10:36:22 PM  >>> |
| **Source Code** |
| # CSCI 2061, Assignment 08, Problem 01  # biniam Lemma  # This program have 4 classes: Time, Date, MilitaryTime, and DateTime  # Main function  def main():  print("Standard time:")  time1 = Time(8,36,0,'PM')  print( time1.getTime() )  time1.showTime();  print()  print("Standard date:")  date1 = Date(13,10,2015)  print( date1.getDate())  date1.showDate()  print()  print("Date time: ")  dt1 = DateTime(8,36,0,'PM',13,10,2015)  print(dt1.getDateTime())  dt1.showDateTime()  print()  print("Military time:")  mt1 = MilitaryTime(2236,22)  print(mt1.getMilitaryTime())  mt1.showMilitaryTime()  mt1.showStandardTime()    # Time class which is base class for MilitaryTime and DateTime  class Time:  # Constructor that takes four arguments and initialize the data  def \_\_init\_\_(self, hour=0, minute=0, second=0, period = 'AM'):  self.\_hour = hour  self.\_minute = minute  self.\_second = second  self.\_period = period  # Accessor method that returns the time  def getTime(self):  return (self.\_hour, self.\_minute, self.\_second, self.\_period)  # mutetor method that takes four arguments and initialize datas  def setTime(self, hour, minute, second, period):  self.\_hour = hour  self.\_minute = minute  self.\_second = second  self.\_period = period  # showTime method that displays the data  def showTime(self):  print(self.\_hour, end=":")  print(self.\_minute, end=":")  print(self.\_second, self.\_period)  # Date class which is base class for MilitaryTime and DateTime  class Date:  # Constructor that takes three arguments and initialize the data  def \_\_init\_\_(self, day=0, month=0, year=0):  self.\_day = day  self.\_month = month  self.\_year = year  # Accessor method that returns the Date  def getDate(self):  return (self.\_day, self.\_month, self.\_year)  # mutetor method that takes four arguments and initialize datas  def setDate(self, day, month, year):  self.\_day = day  self.\_month = month  self.\_year = year  # showDate method that displays the data  def showDate(self):  if self.\_month is 1:  month = "January"  elif self.\_month is 2:  month = "February"  elif self.\_month is 3:  month = "March"  elif self.\_month is 4:  month = "April"  elif self.\_month is 5:  month = "May"  elif self.\_month is 6:  month = "Jun"  elif self.\_month is 7:  month = "July"  elif self.\_month is 8:  month = "Augest"  elif self.\_month is 9:  month = "September"  elif self.\_month is 10:  month = "October"  elif self.\_month is 11:  month = "November"  elif self.\_month is 12:  month = "Desember"  else:  print("invalid month")  print(month, end=",")  print(self.\_day, self.\_year)  # MilitaryTime class that extends Time and Date classes  class MilitaryTime (Time, Date):  # Constructor that takes two arguments and initialize the data  def \_\_init\_\_(self, militaryHours=0, militarySeconds=0):  super()  self.\_militaryHours = militaryHours  self.\_militarySeconds = militarySeconds  # Accessor method that returns the time  def getMilitaryTime(self):  return (self.\_militaryHours, self.\_militarySeconds)  # mutetor method that takes four arguments and initialize datas  def setMilitaryTime(self, militaryHours=0, militarySeconds=0):  self.\_militaryHours = militaryHours  self.\_militarySeconds = militarySeconds  # showMilitaryTime method that displays the data  def showMilitaryTime(self):  print(self.\_militaryHours, self.\_militarySeconds)  # convertToStandard method that converts the militarytime to standard  def convertToStandard(self):  array = []  splited = []  for i in str(self.\_militaryHours):  array.append(i)  x = 1  while x < 4:  splited.append(array[x-1] + array[x])  x+=2  self.\_militaryHour = int(splited[0])  self.\_militaryMinute = int(splited[1])    self.\_period = "AM"  if self.\_militaryHour > 12:  self.\_militaryHour-=12  self.\_period = "PM"  # showStandardTime metod that displays the converted standard time  def showStandardTime(self):  self.convertToStandard()  print(self.\_militaryHour, end=":")  print(self.\_militaryMinute, end=":")  print(self.\_militarySeconds, self.\_period)    # DateTime class extends Date class and Time class  class DateTime(Date, Time):  # Constructor that takes seven arguments and initialize the data  def \_\_init\_\_(self, hour=0, minute=0, second=0, period="AM", day=0, month=0, year=0):    super()  super()  self.\_hour = hour  self.\_minute = minute  self.\_second = second  self.\_period = period  self.\_day = day  self.\_month = month  self.\_year = year  # Accessor method that returns the time and date  def getDateTime(self):  return (self.\_hour, self.\_minute, self.\_second, self.\_period, self.\_day, self.\_month, self.\_year)  # mutetor method that takes four arguments and initialize datas  def setDateTime(self, hour, minute, second, period, day, month, year ):  super(hour, minute, second, period)  super(day, month, year)  # showDateTime method that displays the data  def showDateTime(self):  super().showTime()  super().showDate()    if \_\_name\_\_ == "\_\_main\_\_": main() |

|  |
| --- |
| **Program 02** |
| **Output** |
|  |
| **Source Code** |
|  |

|  |
| --- |
| **Program 03** |
| **Output** |
|  |
| **Source Code** |
|  |

|  |
| --- |
| **Program 04** |
| **Output** |
|  |
| **Source Code** |
|  |

|  |
| --- |
| **Program 05** |
| **Output** |
|  |
| **Source Code** |
|  |