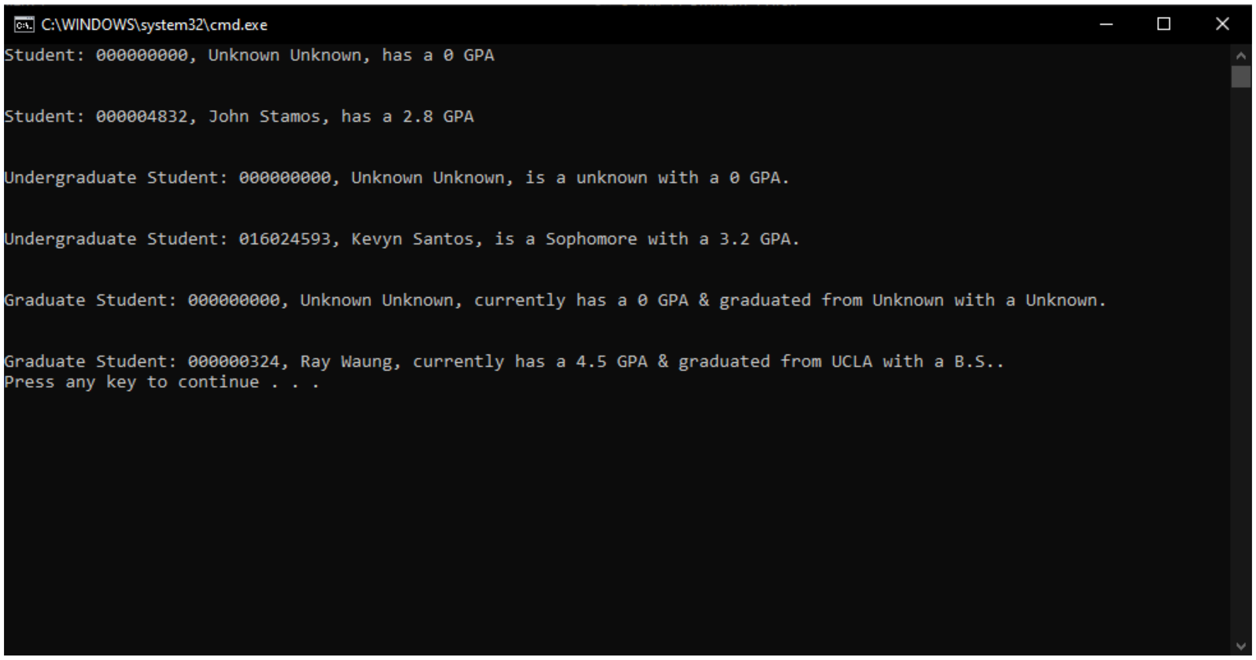
1



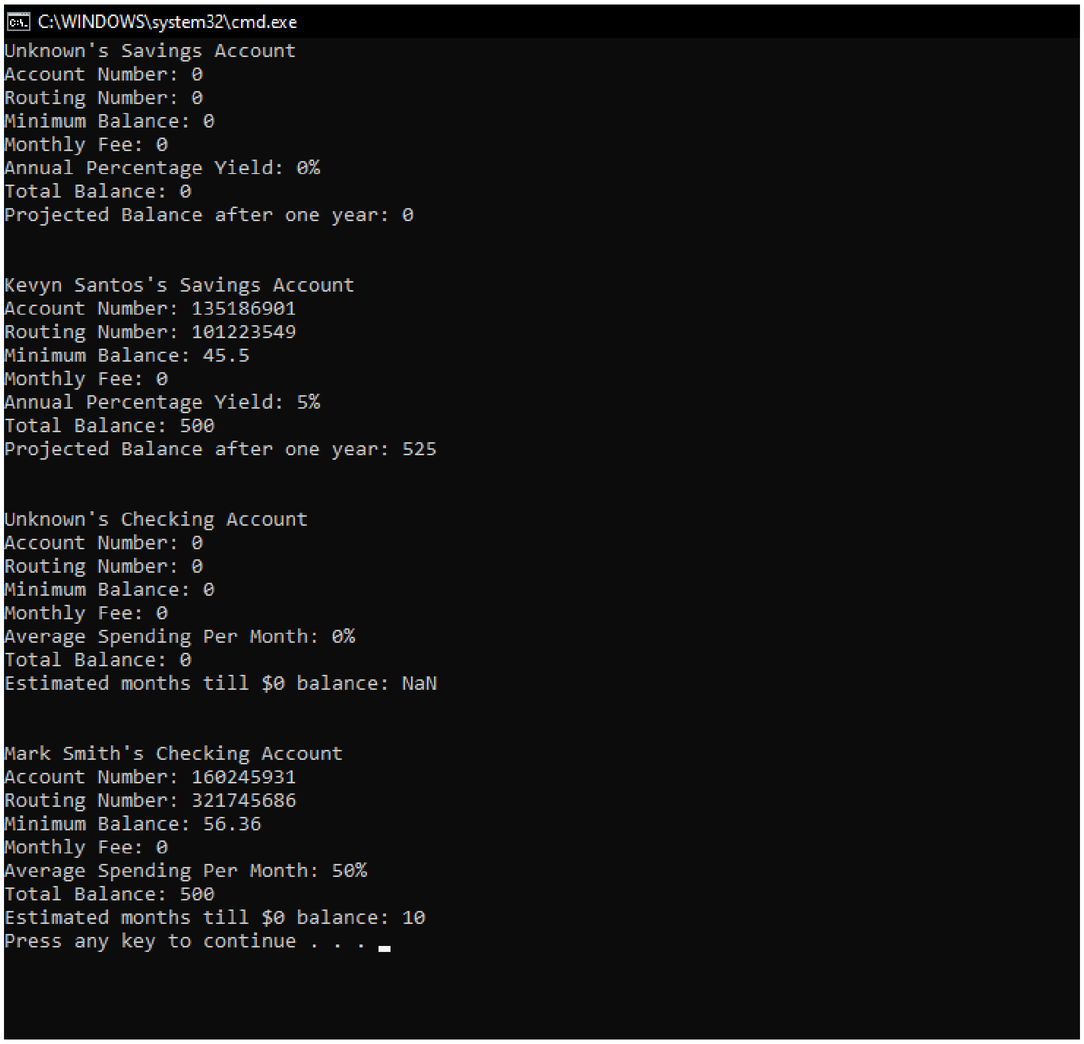
```
Student: 000000000, Unknown Unknown, has a 0 GPA


Student: 000004832, John Stamos, has a 2.8 GPA


Undergraduate Student: 000000000, Unknown Unknown, is a unknown with a 0 GPA.


Undergraduate Student: 016024593, Kevyn Santos, is a Sophomore with a 3.2 GPA.


Graduate Student: 000000000, Unknown Unknown, currently has a 0 GPA & graduated from Unknown with a Unknown.


Graduate Student: 000000324, Ray Waung, currently has a 4.5 GPA & graduated from UCLA with a B.S..
Press any key to continue . . .
```

```
class Student
  {
      protected string firstName;
      protected string lastName;
      protected string studentID;
      protected double gpa;

      public Student()
      {
         firstName = "Unknown";
         lastName = "Unknown";
         studentID = "000000000";
         gpa = 0.0;
      }

      public Student(string f, string l, string studId, double gpA)
      {
         firstName = f;
         lastName = l;
         studentID = studId;
         gpa = gpA;
      }

      public Student(string f, string l)
      {
            firstName = f;
            lastName = l;
            studentID = "000000000";
            gpa = 0.0;
        }

        public Student(string studId, double gpA)
        {
            firstName = "Unknown";
            lastName = "Unknown";
            studentID = studId;
            gpa = gpA;
        }

        public string FirstName
        {
            get { return firstName; }
            set { firstName = value; }
        }

        public string LastName
        {
            get { return lastName; }
            set { lastName = value; }
        }

        public string StudentID
        {
            get { return studentID; }
            set { studentID = value; }
        }

        public double GPA
        {
            get { return gpa; }
            set
            {
                if (value < 0)
                {
                    gpa = 0;
                }
else
                {
                    gpa = value;
                }

            }
        }

        public override string ToString()
        {
            return "Student: " + studentID + ", " + firstName + " " + lastName + ", has a " +
gpa.ToString() + " GPA";
        }

        ~Student() { }

    }







class UnderGrad:Student
    {
        private string classification;

        public UnderGrad() : base()
        {
            classification = "unknown";
        }

        public UnderGrad(string Class, string f, string l, string studId, double gpA) : base(f, l, studId,
gpA)
        {
            classification = Class;
        }


        public string Classification
        {
            get { return classification; }
            set { classification = value; }
        }
        public override string ToString()
        {
            return "Undergraduate Student: " + studentID + ", " + firstName + " " + lastName + ", is a
" + classification + " with a " + gpa.ToString() + " GPA.";
        }

        ~UnderGrad() { }

    }
```

```
class Grad:Student
    {
        private string bachelorType;
        private string institutionofGrad;

        public Grad() : base()
        {
            bachelorType = "Unknown";
            institutionofGrad = "Unknown";
        }

        public Grad(string bachtype, string graduationschool, string f, string l, string studId, double
gpA) :base(f, l, studId, gpA)
        {
            bachelorType = bachtype;
            institutionofGrad = graduationschool;
        }

        public string BachelorsType
        {
            get { return bachelorType; }
            set { bachelorType = value; }
        }

        public string InstitutionOfGrad
        {
            get { return institutionofGrad; }
            set { institutionofGrad = value; }
        }

        public override string ToString()
        {
            return "Graduate Student: " + studentID + ", " + firstName + " " + lastName + ", currently
has a " + GPA.ToString() + " GPA & graduated from " + institutionofGrad + " with a " +
bachelorType + ".";
        }

        ~Grad() { }


    }
```

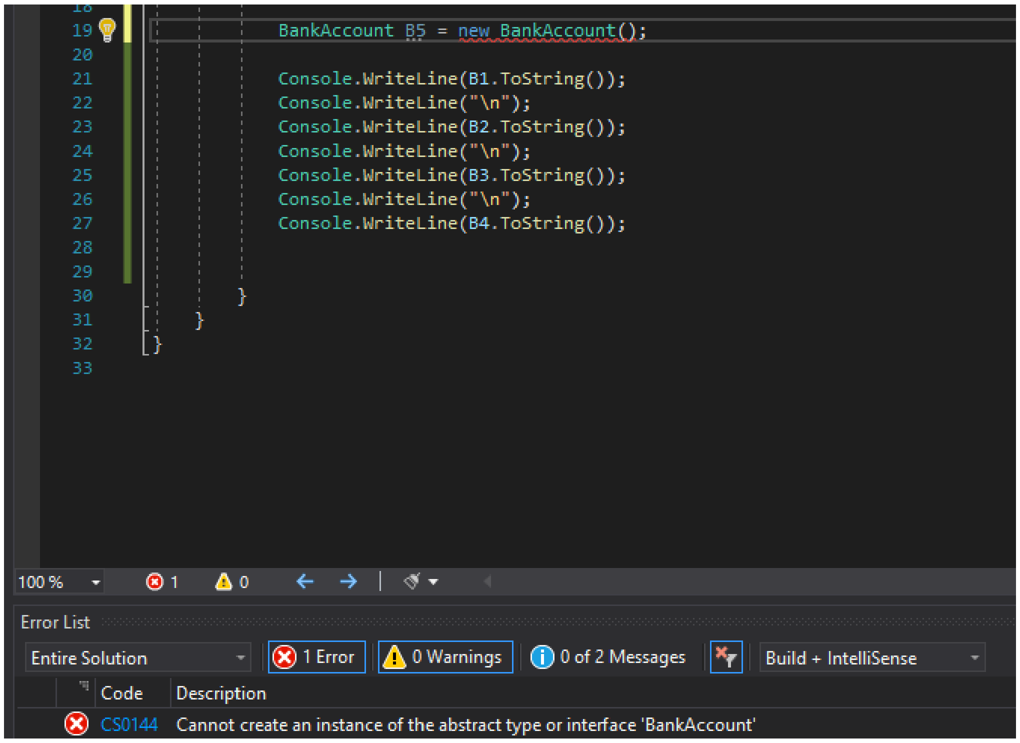
```
class Program
    {
        static void Main(string[] args)
        {
            Student S1 = new Student();
            Student S2 = new Student("John", "Stamos", "000004832", 2.8);

            UnderGrad S3 = new UnderGrad();
            UnderGrad S4 = new UnderGrad("Sophomore", "Kevyn", "Santos", "016024593", 3.2);

            Grad S5 = new Grad();
            Grad S6 = new Grad("B.S.", "UCLA", "Ray", "Waung", "000000324", 4.5);
            Console.WriteLine(S1.ToString());
            Console.WriteLine("\n");
            Console.WriteLine(S2.ToString());
            Console.WriteLine("\n");
            Console.WriteLine(S3.ToString());
            Console.WriteLine("\n");
            Console.WriteLine(S4.ToString());
            Console.WriteLine("\n");
            Console.WriteLine(S5.ToString());
            Console.WriteLine("\n");
            Console.WriteLine(S6.ToString());




        }
    }
```

2



C:\WINDOWS\system32\cmd.exe

```
Unknown's Savings Account
Account Number: 0
Routing Number: 0
Minimum Balance: 0
Monthly Fee: 0
Annual Percentage Yield: 0%
Total Balance: 0
Projected Balance after one year: 0


Kevyn Santos's Savings Account
Account Number: 135186901
Routing Number: 101223549
Minimum Balance: 45.5
Monthly Fee: 0
Annual Percentage Yield: 5%
Total Balance: 500
Projected Balance after one year: 525


Unknown's Checking Account
Account Number: 0
Routing Number: 0
Minimum Balance: 0
Monthly Fee: 0
Average Spending Per Month: 0%
Total Balance: 0
Estimated months till $0 balance: NaN


Mark Smith's Checking Account
Account Number: 160245931
Routing Number: 321745686
Minimum Balance: 56.36
Monthly Fee: 0
Average Spending Per Month: 50%
Total Balance: 500
Estimated months till $0 balance: 10
Press any key to continue . . .
```



```
            BankAccount B5 = new BankAccount();

            Console.WriteLine(B1.ToString());
            Console.WriteLine("\n");
            Console.WriteLine(B2.ToString());
            Console.WriteLine("\n");
            Console.WriteLine(B3.ToString());
            Console.WriteLine("\n");
            Console.WriteLine(B4.ToString());


        }
    }
}
```

```
100 %    1    0
Error List
Entire Solution    1 Error    0 Warnings    0 of 2 Messages    Build + IntelliSense
Code    Description
CS0144  Cannot create an instance of the abstract type or interface 'BankAccount'
```

```
abstract class BankAccount
  {
    protected string holderName;
    protected Int64 accountnumber;
    protected int routingnumber;
    protected double balance;
    protected double minimumBalance;
    protected double monthlyFees;

    public BankAccount()
    {
      holderName = "Unknown";
      accountnumber = 0;
      routingnumber = 0;
      balance = 0;
      minimumBalance = 0;
    }

    public BankAccount(string name, Int64 accnum, int rounum, double bal, double minbal)
    {
      holderName = name;
      accountnumber = accnum;
      routingnumber = rounum;
      balance = bal;
    minimumBalance = minbal;
}

public BankAccount(Int64 accnum, int rounum, double bal, double minbal)
{
    holderName = "Unknown";
    accountnumber = accnum;
    routingnumber = rounum;
    balance = bal;
    minimumBalance = minbal;
}

public string HolderName
{
    get { return holderName; }
    set { holderName = value; }
}

public Int64 AccountNumber
{
    get { return accountnumber; }
    set
    {
        if (value < 0)
        {
            accountnumber = 0;
        }
        else if(value > 999999999999) //account numbers typically 12 digits or less
        {
            accountnumber = 0;
        }
        else
        {
            accountnumber = value;
        }
    }
}

public int RoutingNumber
{
    get { return routingnumber; }
    set
    {
        if (value < 0)
```

```
            {
                routingnumber = 0;
            }
            else if (value > 999999999) //routing numbers typically 9 digits or less
            {
                routingnumber = 0;
            }
            else
            {
                routingnumber = value;
            }
        }
    }

    public double Balance
    {
        get { return balance; }
        set { balance = value; } //no validation for less than 0 because both account types can
have negative balance due to overdraft fees, late fees, etc.
    }

    public double MinimumBalance
    {
        get { return minimumBalance; }
        set
        {
            if (value < 0)
            {
                minimumBalance = 0;
            }
            else
            {
                minimumBalance = value;
            }
        }
    }

    public double MonthlyFees
    {
        get { return monthlyFees; }
        set
        {
            if (value < 0)
            {
                monthlyFees = 0;
            }
            else
            {
                monthlyFees = value;
            }
        }
    }

    ~BankAccount() { }


}
```

```
class SavingsAcc:BankAccount
    {
        private double annualPercentageYield;

        public SavingsAcc() : base()
        {
            annualPercentageYield = 0;
        }

        public SavingsAcc(double APY, string name, Int64 accnum, int rounum, double bal, double
minbal):base(name, accnum, rounum, bal, minbal)
        {
            annualPercentageYield = APY;
        }



        public double AnnualPercentageYield
        {
            get { return annualPercentageYield; }
            set
            {
                if (value < 0)
                {
                    annualPercentageYield = 0;
                }
                else
                {
                    annualPercentageYield = value;
                }
            }
        }

        public double ProjectedBalAftYear()
        {
            double apYmath = annualPercentageYield / 100;
            double BalAftYear = (balance * apYmath)+balance;
            BalAftYear = Math.Round(BalAftYear, 2);
            return BalAftYear;
        }

        public override string ToString()
        {
            return holderName + "'s Savings Account\nAccount Number: " + accountnumber +
"\nRouting Number: " + routingnumber + "\nMinimum Balance: " + minimumBalance +
"\nMonthly Fee: " + monthlyFees + "\nAnnual Percentage Yield: " +
annualPercentageYield+"%\nTotal Balance: " + balance + "\nProjected Balance after one year:
"+ProjectedBalAftYear();
        }

        ~SavingsAcc() { }



    }
```

```
class CheckingsAcc:BankAccount
  {
      private double averageSpendingPrMonth;

      public CheckingsAcc() : base()
      {
         averageSpendingPrMonth = 0;
      }

      public CheckingsAcc(double ASPM, string name, Int64 accnum, int rounum, double bal,
double minbal):base(name, accnum, rounum, bal, minbal)
      {
         averageSpendingPrMonth = ASPM;
      }


      public double AverageSpendingPrMonth
      {
         get { return averageSpendingPrMonth; }
         set
         {
            if (value < 0)
            {
               averageSpendingPrMonth = 0;
            }
            else
            {
               averageSpendingPrMonth = value;
            }
         }
      }

      public double EstMonthsTillBroke()
      {
         double brokemonth = balance / averageSpendingPrMonth;
         brokemonth = Math.Round(brokemonth, 2);
         return brokemonth;
      }
```

```
        public override string ToString()
        {
            return holderName + "'s Checking Account\nAccount Number: " + accountnumber +
"\nRouting Number: " + routingnumber + "\nMinimum Balance: " + minimumBalance +
"\nMonthly Fee: " + monthlyFees + "\nAverage Spending Per Month: " +
averageSpendingPrMonth + "%\nTotal Balance: " + balance + "\nEstimated months till $0
balance: " + EstMonthsTillBroke();
        }


        ~CheckingsAcc() { }

    }
```

```
class Program
    {
        static void Main(string[] args)
        {
            SavingsAcc B1 = new SavingsAcc();
            SavingsAcc B2 = new SavingsAcc(5.0, "Kevyn Santos", 0135186901, 101223549, 500,
45.50);

            CheckingsAcc B3 = new CheckingsAcc();
            CheckingsAcc B4 = new CheckingsAcc(50, "Mark Smith", 0160245931, 321745686,
500, 56.36);


            Console.WriteLine(B1.ToString());
            Console.WriteLine("\n");
            Console.WriteLine(B2.ToString());
            Console.WriteLine("\n");
            Console.WriteLine(B3.ToString());
            Console.WriteLine("\n");
            Console.WriteLine(B4.ToString());


        }
    }
```