Conversion Constructors

Converting Objects

```
class Money {
...
  Money();
...
};

Money money; //1
int amt = 5; //2
money = amt; //3
```

- How can we support this?

Converting Objects

```
class Money {
...
  Money();
  Money(int amount);
...
};

Money money; //1
int amt = 5; //2
money = amt; //3
```

- How can we support this?
- By adding a conversion constructor
  - Here the default constructor is issued for line 1
  - Then in line 3, there is an implicit casting
  - This invokes the conversion constructor

Converting Objects

```
class Money
{
...
  Money();
  Money(int amount);
  Money(Dollar dollar);
  Money(Gold gold);
...
};
```

```
Money money;                //1
int amt = 5;                //2
money = amt;                //3
Dollar dollar(5, 50);       //4
money = dollar;             //5
Gold gold(10);              //6
money = gold;               //7
```

```
class Dollar
{
...
  Dollar(int dollars, int cents);
  int Dollars();
  int Cents();
...
};
```

```
class Gold
{
...
  Gold(int grams);
  int Grams();
...
};
```

- You can have up to one conversion constructors per source type
- This is because conversions constructors take exactly one parameter

Implementation

```
Money::Money()
{
  amt = 0;
}

Money::Money(int amount)
{
  amt = amount;
}
Money::Money(Dollar dollar) //not used
{
  amt = (dollar.Dollars() * 100) + dollar.Cents();
}

Money::Money(Gold Gold)     //not used
{
  amt = 4275 * gold.Grams();
}
```