created: 8/13/1999; small edits 4/24/2006, 07/04/2011, 08/03/2014

# Fill in the Blanks

This exercise reviews the "scope" of variables and parameters. The *scope* of a variable or formal parameter is the section of code that can "see" (can use) the parameter.

**The scope of an instance variable includes each method body (list of statements) and each constructor body.**

1. In the following program skeleton, click on each button where it would be correct to have the statement: `target = 25`

In other words, click on each button where the variable `target` is in scope.

```
class ScopeEg1
{
  int target;

  ScopeEg1()
  {
    ____________ ;
    . . . .
  }

  void aMethod()
  {
    ____________ ;
      . . . .
  }

  void bMethod()
  {
    ____________ ;
       . . . .
  }

}

class AnotherClass
{
  int sum;
  AnotherClass()
  {
    ____________ ;
    . . . .
  }

  void anotherMethod()
  {
    ____________ ;
      . . . .
  }

  void someMethod()
  {
    ____________ ;
       . . . .
  }
}
```

```
class TesterClass
{
  public static void main (String[] args )
  {
    ______________ ;
  }

}
```

---

**"Outsiders" can access instance variables of an object using "dot notation" unless the instance variable is *private*** (or has default access and is in a different package...but ignore this for now.)

2. In the following program skeleton, click on each button where it would be OK to have the statement: `target = 25;`

```
class ScopeEg2
{
  int target;

  . . . .

}

class AnotherClass
{
  int sum;
  ScopeEg2 first = new ScopeEg2() ;

  void anotherMethod()
  {
    first . ______________ ;

    . . . .
  }

  void someMethod()
  {
    first . ______________ ;

    . . . .
  }
}

class TesterClass
{
  public static void main (String[] args )
  {
    first . ________________ ;
  }

}
```

---

3. In the following program skeleton, click on each button where it would be OK to have the statement `target = 25`

```
class ScopeEg3
{
  private int target;

  ScopeEg3()
  {
    ______________ ;
    . . . .
```

```
   }

   void aMethod()
   {
     ______________  ;
       . . . .
   }

   void bMethod()
   {
     ______________  ;
        . . . .
   }
   . . . .

}

class AnotherClass
{
  int sum;
  ScopeEg3 first = new ScopeEg3() ;

  void anotherMethod()
  {
    first . ______________  ;

    . . . .
  }

  void someMethod()
  {
    first . ______________  ;

    . . . .
  }
}

class TesterClass
{
  public static void main (String[] args )
  {
    ScopeEg3 second = new ScopeEg3() ;

    second . _________________  ;
  }

}
```

---

**Formal parameters can only be seen by the body of their own method.**

4. Click on each button where it would be OK to have the statement `System.out.println( data );`

```
class SomeClass
{
  int sum;

  void aMethod( int data )
  {
    ___________________  ;

    . . . .
  }

  void bMethod()
  {
    __________________ ;

    . . . .
  }
}

class TesterClass
{
  SomeClass some;

  public static void main (String[] args )
  {
    some = new SomeClass();
    some.aMethod( 99 );

    ________________ ;

  }

}
```

---

**It is OK for formal parameters in two different methods to use the same identifier.**

5. In the following program skeleton, click on each button where it would be OK to have the statement `sum = data ;`

```
    class SomeClass
    {
      int sum;

      void aMethod( int data )
      {
        __________________ ;

        . . . .
      }

      void bMethod( int data )
      {
        __________________ ;

        . . . .
      }

      void cMethod( int value )
      {
        __________________ ;

        . . . .
      }

    }
```

---

**A local variable can only be seen in the body of its method by statements <u>following</u> its declaration. It is OK for local variables in different methods to use the same name.**

6. Click on each button where it would be OK to have the statement `value = 5;`

```
    class SomeOtherClass
    {
      int sum;

      void aMethod( int data )
      {
        int value;

        ____________________ ;

        . . . .
      }

      void bMethod( int data )
      {
        ____________________ ;

        . . . .
      }

      void cMethod( )
      {
        . . . .

        ____________________ ;

        int value;

        . . . .
      }


      void dMethod( )
      {
        double value;

        ____________________ ;

        . . . .
      }
    }
```

---

**If a local variable has the same name as an instance variable the local variable will be the one seen by the statements in its method that follow its declaration.** (Although it is correct syntax to have both local and instance variables use the same name, it is probably a bad idea since it confuses humans.)

7. Decide if each statements sets the **instance variable** sum or the **local variable** sum.

```
class YetOtherClass
{
  int sum;  // the instance variable

  void aMethod( int data )
  {

    sum = data ;  //____________  ;

    . . . .
  }

  void bMethod( int data )
  {
    int sum;  // a local variable

    sum = data ;  //____________  ;

    . . . .
  }

  void cMethod( )
```

```
    {
      . . . .

      sum = 23 ;  //____________  ;

      int sum;

      . . . .
    }

  }
```

---

**If a local variable has the same name as an instance variable and you want to specify the instance variable, use *this.***

8. In the following program skeleton, decide if each statements sets the **instance variable** sum or the **local variable** sum.

```
  class AfurtherClass
  {
    int sum;  // the instance variable

    void aMethod( int data )
    {
      int sum;  // a local variable

      this.sum = data ;  //____________  ;

      . . . .
    }

    void bMethod( int data )
    {
      int sum;  // a local variable

      sum = data ;  //____________  ;

      . . . .
    }

  }
```

---

**If a *parameter* has the same name as an instance variable and you want to specify the instance variable, use *this.*** This is often done with constructors, where it is probably *less* confusing to use the same name for both.

9. Decide if each statement sets the **instance variable** sum or the **parameter** sum.

```
  class AfurtherClass
  {
    int sum;  // instance variable

    AfurtherClass( int sum )  // constructor
    {
      this.sum = sum ;  //____________  ;

      . . . .
    }

    void bMethod( int sum )
    {

      sum = 32 ;  //____________  ;
      . . . .
    }

  }
```

---

**An "outsider" can change a *private* instance variable of an object by using an access method of the object (if there is one.)**

10. In the following program skeleton, click on those buttons next to statements that change *sum*.

```
  class SimpleClass
  {
    private int sum;

    SimpleClass( int s )
    {
      sum = s ;  //____________  ;

      . . . .
    }

    void setSum ( int s )
    {

      sum = s ;  //____________  ;

      . . . .
    }

  }

  class TesterClass
  {
    public static void main ( String[] args )
    {
      SimpleClass sim = SimpleClass( 34 );  //____________  ;

      sim.sum = 77 ;  //____________  ;

      sim.setSum( 14 ) ;  //____________  ;
    }
  }
```

---

End of the Exercise. If you want to do it again, click on "Refresh" in your browser window.