**Separate Classes:**

**Solution 1.**

**i)** Objects contain group of variables

Eg; point (int x, int y) , Date (int dd, int mm, int yyyy).

**ii)** We require template classes for each object template class to describe how to make an object. Once the template is described, we can get java to make objects of that class at run time.Hence, we say objects are instances of class.Template class from which objects are obtained.

**iii)** This is used as a reference to the object that is being created by the constructor or that owns the instance method.

**iv)** Date is the object or instance of class date which contains variables day, month and year.

**v)** Constructor method creates an object and returns a value, which is a reference to the newly created object.

In Constructor no return type value is defined and the method name is the same as the class name.

**vi)** The similarity in the notation for accessing the class methods or instance variables is accessing them through objects.

Eg; point is a template class.

P1 is an object.

For accessing either method or variable we have to create object and their access

P1.method / p1. Variable

**Solution 2.**

**i)** Yes, it's legal.

Output- The two points are equivalent.

**ii)** Output- (15, 15)

**iii)** In the Constructor method no return date type is defined.

**iv)** Class methods are executed in a static context that means it can be used as soon as the class is loaded in a virtual machine.

If we omit static or when we declare class method in dynamic context it means there are as many copies of the code at runtime as there are instances. Instance methods belong to an instance.

**v)** When we need to print spaces or we use many methods in the class template to do a particular task without passing parameters.

**vi)** Printing using instance method toString().

**Solution 3.**

**i)** toString().

**ii)** No, there are no such types which are both primitive and reference.

String is a reference type.

**Solution 4.**

**i)** Class variables are declared in static context while instance variables are declared in dynamic context.

**ii)** It is always a better practice to make instance variables private. This permits us to alter the way we implement the class, without it affecting the code in other classes.

We can also store the history of how many times it has been set.

We can also control the setting of variables.

**iii)** Date birthDate;{ = new Date( constructor method arguments)};

**iv)** Int date month year.

**v)**

**1)** Variable is an object.

**2)** A variable contains an object.