## Memory Model

- All variables are allocated space in memory to store something.
- Java divides the computer's memory into two parts: a stack and a heap.
- Every method has a stack frame with space for all its local variables and parameters.
- Objects are stored in the heap.
- If a local variable/parameter is a primitive, the data is stored in the stack. If a local variable/parameter is an object, the stack stores a reference to the object in the heap.
- Fields reside in the objects on the heap.
  - If a field is a primitive type, the value is stored directly in the object.
  - If a field is an object type, it stores a pointer to another object on the heap.

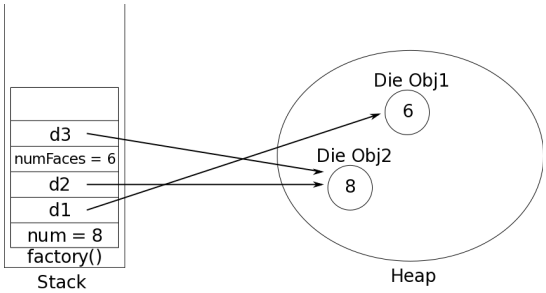
```
// suppose num is 8
public void factory (int num) {

      Die d1 = new Die();

      Die d2 = new Die(num);

      int numfaces = d1.getFaces();

      Die d3 = d2;
}
```




## Variable, Scope, Lifetime, and Default Values

- There are two kinds of variables: instance variables/fields and local variables/parameters.
  - Instance variables are what we call fields that store the data of objects.
  - Local variables are the temporary variables defined inside a method. Parameters are just local variables in Java except that they are initialized by the caller instead of in the method body.

- Local variables' lifetimes consist of the time the method that contains them starts until the method returns.
  - Q: `d1`, `d2`, and `d3` are local variables in the above code. If we add another line "`d1 = d2;`" after the last line, what will happen?
    - `d1`, `d2`, and `d3` will point to `Die Obj2`, and `Die Obj1` is garbage collected.
- Fields' lifetimes consist of the time from the creation the object until the garbage collection of the object.

- Local variables' scope is from the point of declaration to the end of the enclosing scope.
- Fields' scope depends on whether they are public or private. The scope of private fields is the whole class body in which the fields are declared. The scope of public fields is everywhere.

| | local variables/parameters | instance variables |
|---|---|---|
| **location** | stack frame | heap |
| **lifetime** | during method execution | object creation to garbage collection |
| **scope** | remaining body of method | class if private, all if public |

- We use `new` command to create an object. This command allocates memory for a new object, including the fields of the object, clears out that memory, calls the constructor to initialize the fields, and then returns a pointer to the memory.

- It's not always necessary to assign a value when a field is declared. Fields that are declared but not initialized will be set to a default value by the compiler. The default value will be zero or null, depending on the data type. We do not recommend to rely on the default values. (boolean is false, String/object is null, and other primitives are zero).
- Local variables must be initialized before using. Compiler never assigns a default value to an uninitialized local variable. Accessing an uninitialized local value will result in a compile-time error.

REF: [https://docs.oracle.com/javase/tutorial/java/nutsandbolts/datatypes.html]

---

## Static Fields and Methods

- Methods in a class are usually there to manipulate the data in the fields for you (e.g., get an set them). But what if a class has no fields? Then the objects of the class all do exactly the same thing, so there is never a reason to create more than one object of that class.
- To avoid having to create any objects of the class in that situation, Java has the keyword "static".
- Static methods belong to a class, not to an object. That is, they are things you ask classes to do, not objects—when main started, there was a class Die but there was not Die object created yet. We asked the class Die to execute the main method.
- Where else have we seen static methods? [Math.random]
- Can have static fields as well that belong to the class. All objects of that class have access to the static variable—they can all modify the same variable; they don't have their own copies of it like instance fields. Where have we seen static fields? [System.out]