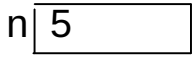
Program Memory
Recall in Java the 8 primitive types: int, short, long, byte, float, double, boolean, char

In Java, a primitive type is a value type. This means it stores its value directly at its memory location.

In Java, a type defined by a class is a reference type. This means it stores a reference to another memory location where the actual object is.

Car c = new Car();

int n = 5;



The two main sections of a program's memory are the program stack (call stack or stack) and the program heap (heap).

The stack is composed of stack frames.
A stack frame is added to the top of the stack when a method is called.
The stack frame is removed from the top of the stack the method returns.

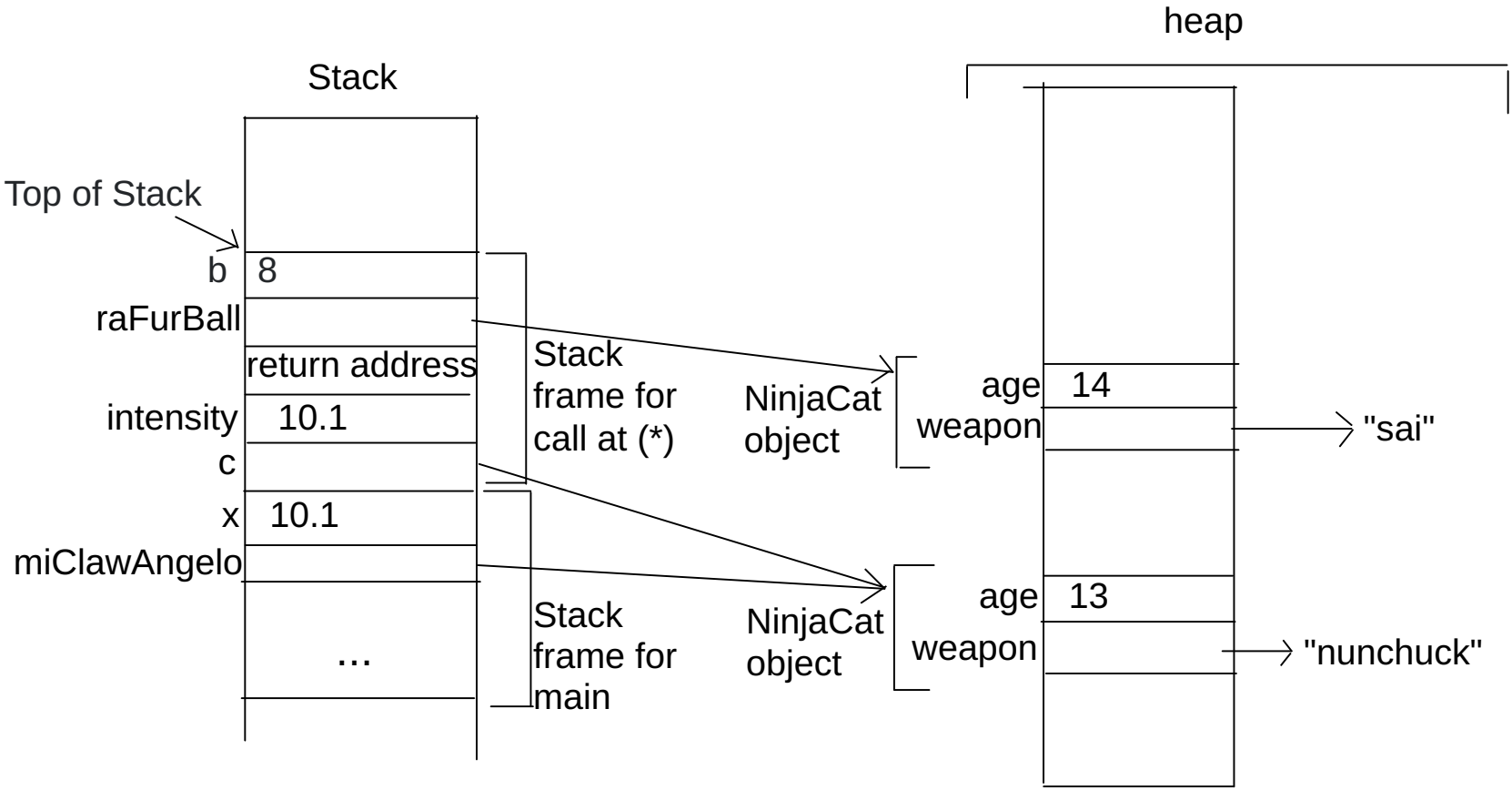
The heap is used for dynamic data that needs to be created on the fly at runtime.
The keyword new allocates data on the heap.
new returns a reference to memory allocated on the heap.

```
public class NinjaCat
{
   private int age;
   private String weapon;

   public NinjaCat(int theAge, String theWeapon)
   {
      age = theAge;
      weapon = theWeapon;
   }

   public void yeet(String obj, double intensity)
   {
       // ...
   }

   public void setAge(int newAge)
   {
      age = newAge;
   }
}
```

```
public class FightScene
{
  public static void main(String[] args)
  {
    NinjaCat miClawAngelo = new NinjaCat(13, "nunchuck");
    double x = 10.1;
    foodFight(x, miClawAngelo); // method call (*)

    // (2)
    // ...
  }

  public static void foodFight(double intensity, NinjaCat c)
  {
    NinjaCat raFurBall = new NinjaCat(14, "sai");
    raFurBall.yeet("pizza", intensity);
    int b = 8;

    // (1)

    intensity = 2.5;
    c.setAge(15);
  }
}
```

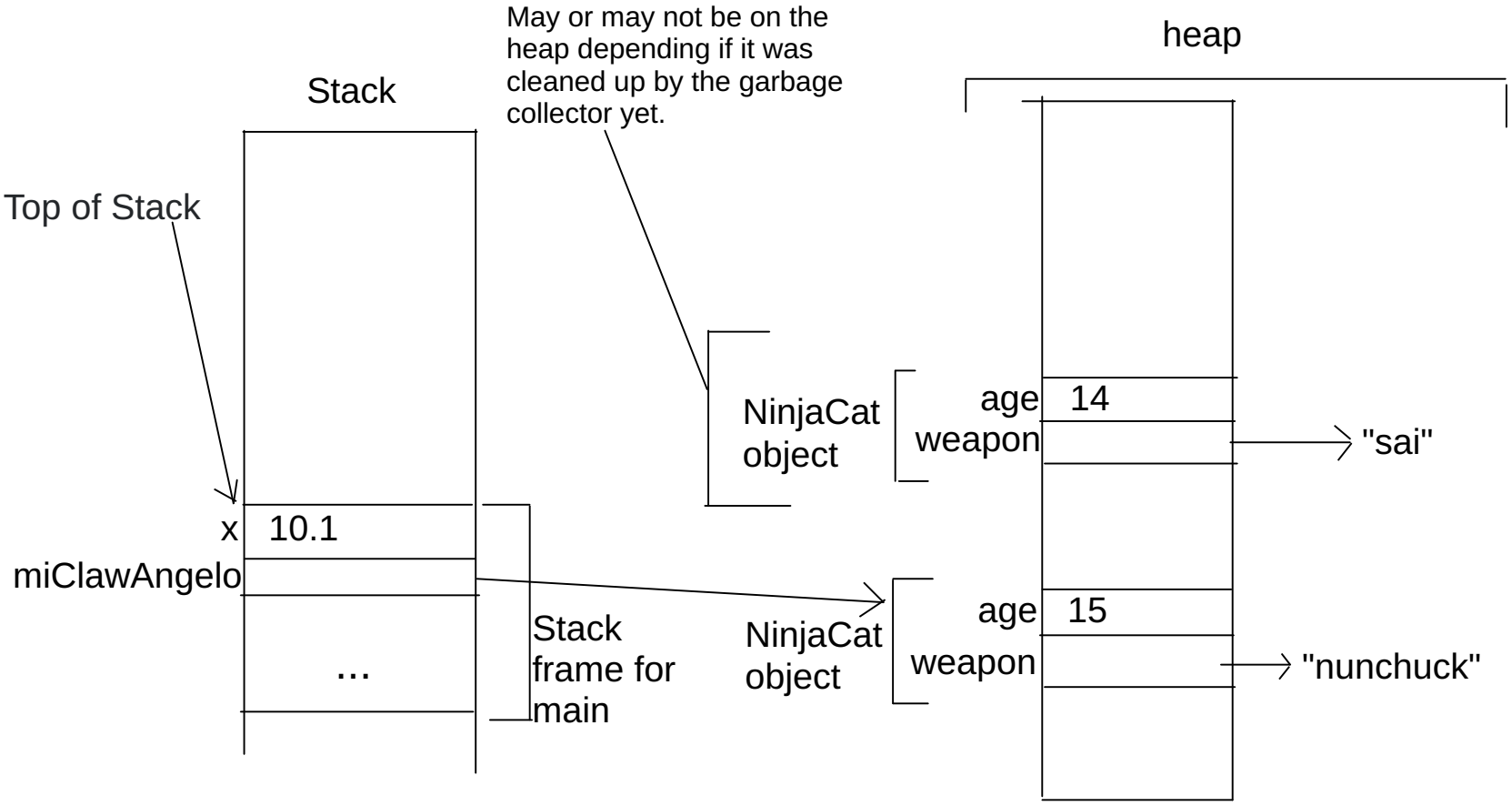
PC = program counter (keeps track of the next line to run when executing your program)

In the previous program, when PC is at (1) from the method call at (*), the memory looks as follows (somewhat simplified):



The return address tells the program where in the code to go back to when the method returns.

In the previous program, when PC is at (2), the stack frame from the call at (*) is popped off the stack and the memory looks as follows:



Notes:
- At (*) x is passed to foodFight by value as parameter intensity, so changes to intensity in foodFight do not change the value x in main
- At (*) miClawAngelo is passed to foodFight by reference as parameter c, so changes to the object c references in foodFight are reflected (seen) in miClawAngelo in main since they both reference the same object on the heap.

Primitive types in Java are passed by value.
Objects in Java are treated as reference type and are passed by reference.

In Java, data that is no longer in use on the heap is eventually removed (reclaimed) automatically by the garbage collector. The garbage collector is built into the JVM.

Note: if an object on the heap still has a reference to it, then the garbage collector will not remove it even if it is not used. Therefore, you should set references to null if they are not being used anymore*.

* When a reference is popped off the stack, it is not considered to be referencing its object anymore. So, in the previous example we do NOT need to (and should not) set raFurBall to null before returning from foodFight.

Wrapper classes for Primitive Types
A primitive type is a value type.
But sometimes we need to treat a primitive type as a reference type.
Some reasons for this:
- To be able to represent the abscense of a value with null.
- The program may need to think of a primitive type as an object (e.g., elements in an ArrayList must be reference types (not primitive types) so as to allow uniform treatment of data types).

A wrapper class is a class that encapsulates the functionality of some other data type or component.

For each of the 8 primitive types, Java has a built in wrapper class.

Primitive Type	Wrapper class
int	Integer
double	Double
char	Character
long	Long
boolean	Boolean
short	Short
byte	Byte
float	Float

Example

```
Integer a = new Integer(7);
// Same as doing:
//    Integer a = 7;
```