|  |  |
| --- | --- |
| 4 Pillars of OOP  Hint: it’s a pie  (Functional Programming vs OOP) | Abstraction: hides the implementation details  Encapsulation: data hiding  Inheritance: let one object acquire properties and methods of another. For reusability  Polymorphism: Many shapes, when you run code it will do things on what object you pass. Think of fire alarm |
| Objects (Heap) | Heap  -area memory (like a big lake) where objects live  -used for dynamic memory allocation of objects and JRE classes at runtime  -Reference is like a pole (stored in “stack” memory) |
| Strings (Identity and Equality) | -only objects created without using new  -optimization to avoid creating too many strings  Identity  -same object in memory  Equality  -contain equal data  Primitives only have equality |
| Constant vs Immutable | Constant  -applies to variables including object reference  -final Alarm alarm1 //only reference is final  Immutable  -applies to objects (Strings)  -cannot change the state of the object |
| Autoboxing | A black background with white text and blue numbers  Description automatically generated  -BTS, Java does it AUTOmatically for you (AUTOboxing)  -Integer is the wrapper class because you wrap it around an int |
| Garbage Collector | -only about memory and not resources (like opening a file/network socket)  -JVM tracks reference for us so if no object has a reference then GC will delete it for us  -Therefore, no destructors needed  -Unpredictable, not ideal for strict runtime computations |
| Access Modifiers  Extra:  Accessor Methods (getter/setters) | Class  -public: visible to code in other packages  -package private: only in its package  Field/Methods/Constructors  -if no public/private then “package private”  More:  A green sign with white text  Description automatically generated |
| Super  final | -super.METHOD to call methods from super  A purple background with white text  Description automatically generated  A close-up of a logo  Description automatically generated |
| Single Rooted Hierarchy | -Every class extends from Object  -All Java classes have ONE direct superclass  -toString() comes from Object |
| Upcasting | -Converting a subclass Object to a superclass reference  -Casting happens on REFERENCE, not on Object  -When Upcast, we cannot call Cat methods  -Like breaking a TV remote, and replacing it with generic basic one |
| Downcasting | -To avoid ClassCast Exception  A blue background with white text  Description automatically generated |
| static  A screen shot of a computer program  Description automatically generated  -static imports, don’t have to use MATH | A blue sign with white text  Description automatically generated  -might be a good idea to make it final  -don’t have to use Class.staticMethods if in the same class  Static block  -only once and executed before constructor |