Lec 15

Classes, Memory
Car

- `typeOfCar`: String
- `color`: String
- `maxSpeed`: int
- `damageLevel`: int

+ `accelerate()`: void
+ `decelerate()`: void
- `blowUp()`: void
+ `takeDamage(damage:int)`: void
public class GTA {
    public static void main(String[] a) {
        Car audi = new Car();
        audi.<what methods are available>?
    }
}
To compile and run GTA.java and Car.java

javac GTA.java Car.java
java GTA

OR

javac GTA.java
java GTA

OR

javac Car.java
javac GTA.java
java GTA
Car audi = new Car();
Car honda = new Car()
Car bmw = audi;

audi == honda;
audi == bmw;
honda == bmw;

a) T T T
b) T T F
c) T F T
d) F T F
e) F F F
public class GTA {
    public static void main(String[] a) {
        Car audi = new Car();
        audi.maxSpeed = 100;
        rocketBooster(audi);
        System.out.println(audi);
        System.out.println(audi.maxSpeed);
    }
    public static void rocketBooster(Car c) {
        c.maxSpeed += 10;
    }
}

Assume Car.maxSpeed is public
public class GTA {
    public static void main(String[] a) {
        Car audi = new Car();
        audi.maxSpeed = 100;
        rocketBooster(audi);
        System.out.println(audi);
        System.out.println(audi.maxSpeed);
    }
    public static void rocketBooster(Car c) {
        c.maxSpeed += 10;
    }
}
this
• Refers to “this instance” of an object
  – Think of as this = object’s memory address

• Used for:
  1. Disambiguate variable references
  2. Call alternate constructors from within a constructor
  3. Pass the current class instance as an argument to a method of another object
public class Car{
    private String color;

    public Car(String color) {
        this.color = color;
    }
}

public class Car {
    private String color;

    public Car() {
        this("white"); // set default color
    }

    public Car(String color) {
        this.color = color;
    }
}
public class FBUser{
    private ArrayList<FBUser> friends;

    public FBUser(){
        friends = new ArrayList<>();
    }

    public void acceptFriend(FBUser newFriend) {
        addFriend(newFriend);
        newFriend.addFriend(this);
    }

    public void addFriend(FBUser newFriend) {
        friends.add(friend);
    }
}
public class ChalkBox {
    private Chalk piece;
    public ChalkBox() {
        piece = new Chalk("green", this);
    }
    public void useChalk(String word) {
        piece.write(word);
        return;
    }
}

public class Chalk {
    private ChalkBox box;
    private String color;
    public Chalk(ChalkBox box) {
        this("white", box);
    }
    public Chalk(String color, ChalkBox box) {
        this.color = color;
        this.box = box;
    }
    public void write(String word) {
        System.out.println("/" + color + "/ " + word + " from box: " + box);
        return;
    }
}

public class Example {
    public static void main(…) {
        ChalkBox c = new ChalkBox();
        c.useChalk("hi");
    }
}

A) Compiler error
B) /white/ hi from box: <address>
C) /green/ hi from box: null
D) /green/ hi from box: <address>
E) None of the above
Memory
Memory

• Static: allocated once and last for the duration of a program
• Stack: region where a method's local variables are allocated during a method call
• Heap: region where the "new" operator allocates memory for objects
Stack

• Method call adds local vars and params to stack
• Return removes vars from stack
• Last In First Out (LIFO)
public class Ex {
    public static void main(...) {
        int i = 0;
        char c = 'A';
        foo(i);
    }
    public static void foo(int k) {
        double d = k * 2;
        bar();
        return;
    }
    public static void bar() {
        int z = 10;
        System.out.println(z);
        return;
    }
}
Draw the order of methods on the stack up to the indicated point of program execution, method6. Note: the stack may not fill completely, also Stacks fill from the bottom to the top:

```
public class Driver {
    public static void main(String[] args) {
        SomeClass sc = new SomeClass();
        sc.method1();
        sc.method4();
    }
}
public class SomeClass {
    public SomeClass() {
    }
    public void method1() {
        method5();
    }
    public void method2() {
        return;
    }
    public void method3() {
        method6();
    }
    public void method4() {
        method2();
    }
    public void method5() {
        method3();
    }
    public void method6() {
        /* What is the stack here? */
    }
}
```
public class Ex {
    public static void main(…) {
        int[] array = new int[10];
        int value = 32;
        Object o = new Object();
    }
}
Heap

• Automatically managed by Java Garbage Collector
  – Keeps track of variables referring to object. Once it gets to 0, mark for deletion