public class Baby {

  private static int count = 0;
  private String name;

  public Baby(String theName) {
    count++;
    name = theName;
  }

  public static int getCount() {
    return count;
  }

  public static void main(String[] args) {
    Baby baby1 = new Baby("Arjun");
    System.out.println(Baby.getCount());
    Baby baby2 = new Baby("Candy");
    System.out.println(Baby.getCount());
  }
}

What does the program output?

A. 1 then 1  
B. 1 then 2  
C. 2 then 2  
D. Compiler error  
E. Whatever
Administrative Details

• Let me know if you haven’t picked up your course packet
• Piazza Sign Up
  • Everyone’s been invited, but not every has signed up
• First Lab today!
  • Prelab
    • Set up accounts and submit google form (if you haven’t already!)
    • Complete Lab 1 design doc
Agenda

Lab 1

- Static variable & method
- toString()
- equals()
- import
Lab1: Silver Dollar Game

- 2 Player Game
- Players take turns and move a coin left
  - Restrictions:
    - Each square holds at most 1 coin
    - Coins can be moved any # of squares, but can’t “jump over” other coins.
- The last player to move wins!
Lab 1: Silver Dollar Game

Variable Types

- Primitive Types: boolean, char, byte, short, int, long, float, double
- Objects: extend Object
  - arrays Strings[ ] args
    - Holds values of a single type
  - (class-based) Objects
    - Can hold information (fields)
    - Can specify behaviors (methods)

```java
public class Baby {
    private String name;
    private int age;

    public Student(int theAge, String theName) {
        age = theAge;
        name = theName;
    }

    public String getName() { return name; }
    public int getAge() { return age; }

    public void setName(String newName) {
        name = newName;
    }

    public void setAge(int newAge) {
        if (newAge > 0) {
            age = newAge;
        } else {
            age = 0;
        }
    }
}
```
Lab 1: Silver Dollar Game

class CoinStrip

Instance Variables

- int[] location // [0, 2, 3, 6]
- boolean[] coinstrip // [t, f, t, f, f, t, f, t, f, t]
- int[] gaps // [0, 1, 3, 2]

Constructor

CoinStrip()
CoinStrip(...)

Method

move Coin

is Over
display (= toString())
Agenda

• Lab 1
  - Static variable & method
    • toString()
    • equals()
    • import
Static variables are shared by all instances of class

Any variable shared by all instances should be declared `static`
- E.g. nursery name, baby count

Any constant (variable shared by “the universe”) should be declared `static final`
- E.g. static final double PI = 3.141592653589793;

value cannot be changed
static methods

- Static methods are shared by all instances of class & can only access static variables and other static methods
- Any method that does not access instance variables should be declared “static”
  - E.g. Baby.getCount(), Math.pow(), Integer.parseInt()
Agenda

• Lab 1
  • Static variable & method
    ○ toString()
  • equals()
  • import
Every class has a `toString` method

“`toString()`", **if correctly implemented**, returns a String representation of the given object.

(Notes, `System.out.println(someObject)` automatically calls `someObject.toString()` unless `someObject == null`.)

E.g.:

```java
Baby b1 = new Baby(18,"Teresa");
//print “Teresa(age 18)”
System.out.println(b1.toString());
//print “Teresa(age 18)”
System.out.println(b1);
```
Agenda

• Lab 1
  • Static variable & method
  • toString()
  • equals()
• import
• `==` checks whether 2 names refer to same object (memory address)
• Every class has a “equals” method
• “equals()”, if correctly implemented, checks whether the contents are the same

E.g.:
```java
Baby b1 = new Baby(18,"Teresa");
Baby b2 = new Baby(18,"Teresa");
System.out.println(b1 == b2); // false
System.out.println(b1.equals(b2)); // true
System.out.println(b2.equals(b1)); // prints true
```
equals() Example

• Defining equals()
  • Check the object is of the same type
  • Check the contents are the same
Agenda

• Lab 1
  • Static variable & method
  • toString()
  • equals()

• import
• “import” allows to refer to classes which are declared in other packages, e.g. Random.

Example: Count # of flips until “heads”

```java
import java.util.Random;

Random rng = new Random();
int flip, count = 0;
flip = rng.nextInt(2); // returns 0 or 1
count++;
while (flip == 0) {
    flip = rng.nextInt(2);
    count++;
}
```

VS

```java
do {
    flip = rng.nextInt(2);
    count++;
} while (flip == 0);
```