Objects Round 1

# AP Computer Science

# Types in Java

# Java Types

- There are two classifications of types in Java
- Primitive types:
  - `int`
  - `double`
  - `boolean`
  - `char`
- Object types:
  - `String`
  - arrays
  - An infinite number of others…

# Characteristics of Primitive Types

Primitive types:

- Have a fixed amount of storage
- Think of them as a box designed to hold a particular kind of data
- Have basic operations for manipulation
  - **`int`**, **`double`** (**`+`**, **`-`**, **`*`**, **`/`**, **`%`**)

# Characteristics of Object Types

Object types

- Hold arbitrarily complex data of any kind
- Do not have a pre-specified amount of storage
- Think of them as arrows pointing to some concrete *thing* containing primitive data
- Use methods for interaction instead of operators
  - For example, **`String`** objects use **`equals()`**, **`compareTo()`**, **`indexOf()`**, etc.

# Parts of an Objects

- Objects consist of two things:
  - Data - attributes used to describe the objects
  - Methods - actions either the object can take or that can be performed on the object

# Parts of an Objects

Example of data and methods for a person

- Data
  - name
  - age
  - hair color
- Methods
  - walk()
  - talk()
  - eat()

# References

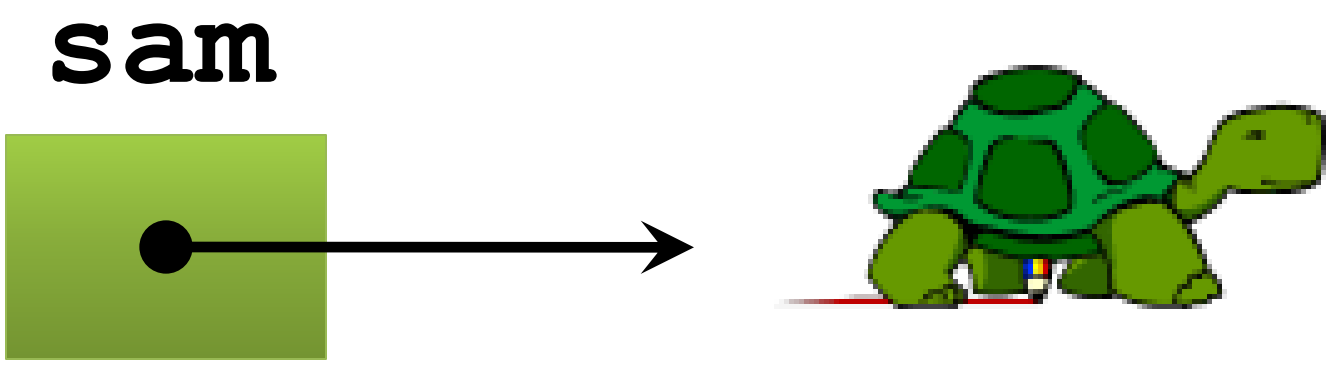
# Reference Types are Different

- Variables holding object types are called **reference variables**
- A primitive variable holds a value
- A reference variable merely references the location of the object

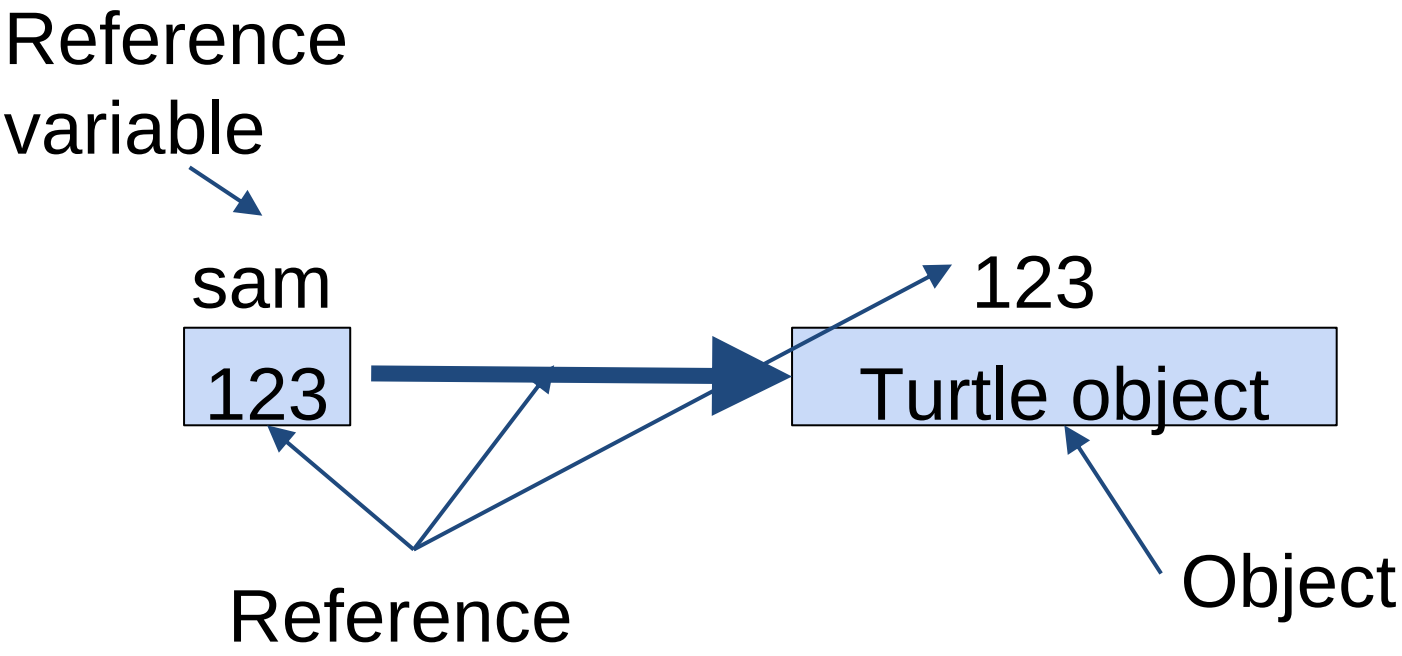
# How does this Affect You?

- Picture a turtle…
- Imagine this turtle is a Java object
- You want a **reference variable** of type **`Turtle`** to point at this specific turtle
- We will call the **reference variable `turtle1`**

```
Turtle sam = new Turtle();
```



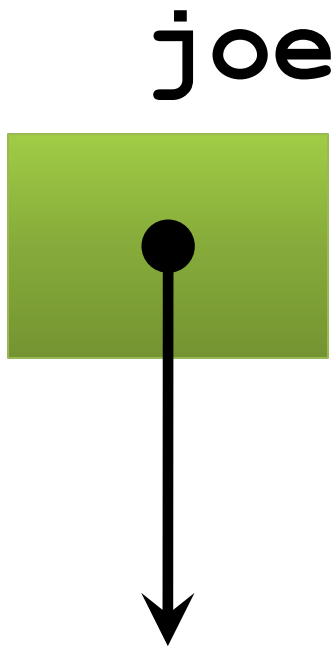
# Initializing a Turtle



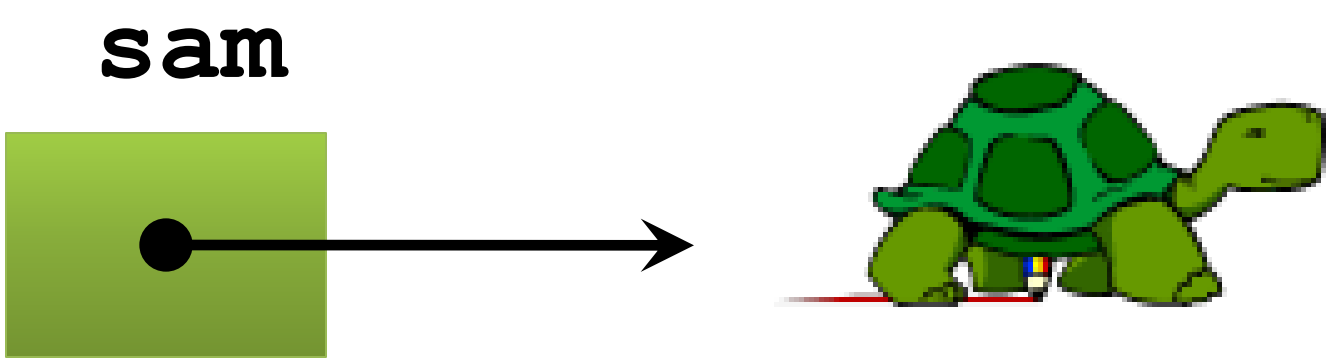

- For a String the identifier is called a **reference variable** which stores a **reference** to a location in memory where the String is located.

# How Many Turtles?

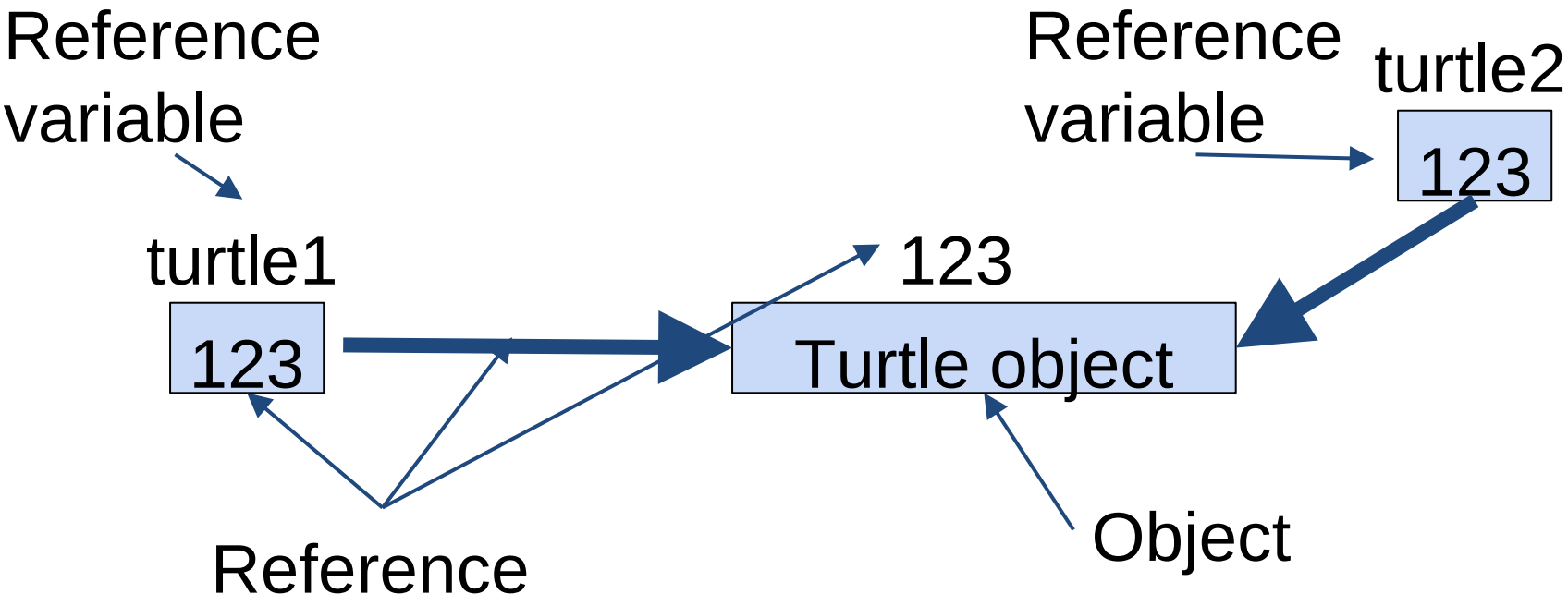
- What if we have another **`Turtle`** reference variable called **`joe`**
- What happens if we set **`joe`** to have the same reference as **`sam`** using the following code?




```
Turtle joe = sam;
```



# Initializing a Turtle




- For a String the identifier is called a **reference variable** which stores a **reference** to a location in memory where the String is located.

# There is Only One Turtle!

- When you assign a reference variable to another reference, you only change what it points to
- This is different from primitive types
- When you do an assignment with primitive types, you actually get a copy

```
int x = 37;
int y = x;
```

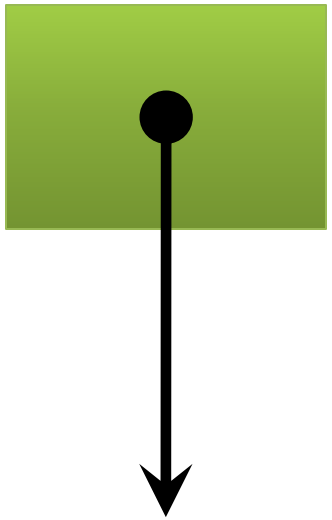
# Reference vs. Primitive Variables

- A reference variable is only a **reference** to a real object
- As we have seen, an object can have more than one name (reference variable)
- These names are called **aliases**
- Any of the references can be used to modify the object

# Turtle Solo

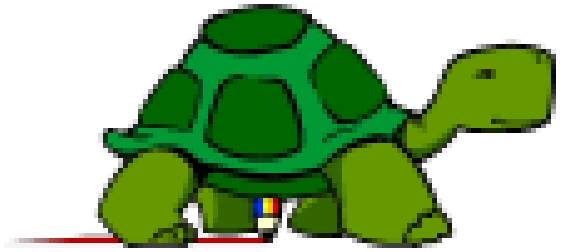
- Thus, if we tell **`turtle2`** to move forward, it will affect the turtle pointed at by **`turtle1`**
- Remember, they are the same turtle

**`turtle2`**



```
turtle2.forward();
```

**`turtle1`**



# Remember, Primitives Make Copies

- We have **`int`**s **`x`** and **`y`**, both with value **`37`**
- If we change **`x`**, it only affects **`x`**
- If we change **`y`**, it only affects **`y`**

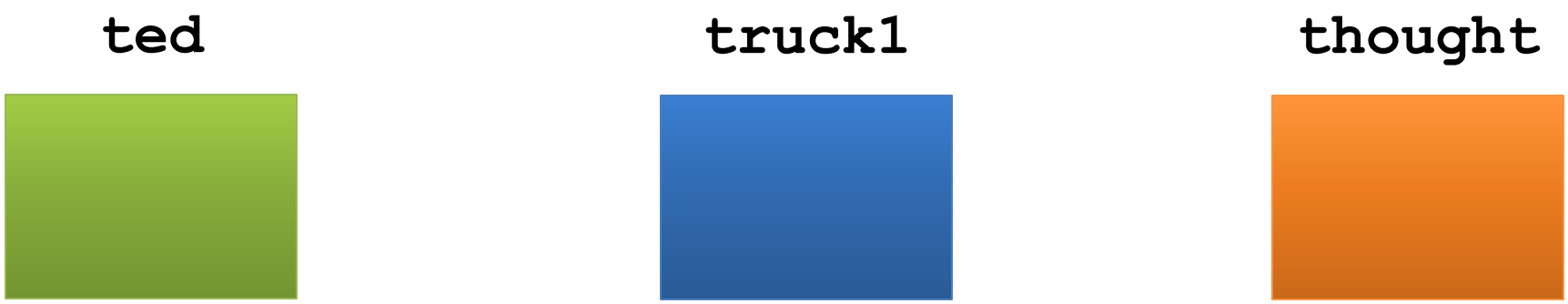
```
int x = 37;
int y = x;
x++;
y--;
```

# Creating New Objects

# A Reference is an Arrow

- If you declare multiple reference variables, you have not created any objects, only the reference variables

```
Dog ted;
DumpTruck truck1;
Idea thought;
```



# Where do those Arrows Point?

- When you first declare a reference variable, those arrows point to **`null`**
- **`null`** is a Java keyword which means nothingness
- If you try to do something with **`null`**, thinking it is a real object, you can break your program

# Constructors

- To make those arrows point at a new object, you must call a **constructor**
- A constructor is a special kind of method used to create an object and the reference
- A constructor will have the same name as the class

# Turtle Constructors

```
public class Turtle { //main in 2nd file
  //instance variables
  private int posX, posY; //must be private
  //default constructor
  public Turtle(){
    posX = 0;
    posY = 0;
  }
  //initialization constructor
  public Turtle(int x, int y){
    posX = x;
    posY = y;
  }
}
```

# `new` Keyword

- Think about the two ways to create Strings, you will remember one using the **`new`** keyword
- Using the **`new`** keyword with a constructor call creates an **object** and a **reference**
- Think of the **`new`** keyword as the “birth” of an object

```
new Scanner(System.in);
new String("hello");
```

# Calling the Default Constructor

- To call a constructor, you use the **`new`** keyword with the name of the constructor followed by parentheses:

```
//calling the default constructor
Turtle turtle1 = new Turtle();
```

- This is an example of calling a default constructor
- You can identify a default constructor because it will have no parameters in the **`()`**

# Calling the Initialization Constructor

- An initialization constructor sets the default values of the instance variables to the value of the parameters passed in
- There is a **`Turtle`** constructor that lets you take an x and y position that is the location of the turtle, and a world for the turtle to be placed in

```
//calling the initialization constructor
World earth = new World();
Turtle turtle2;
turtle2 = new Turtle(100,100,earth);
```

# Calling Methods on Objects

# Object methods

- Object methods are those methods called on objects
- They are different from static methods because they can use information inside the object to perform some task
- Think of them as asking an object a question (for value returning methods) or telling an object to do something (for void methods)

# Calling methods

- You are already familiar with calling methods on **`Strings`**
- Objects are the same!
- Simply type the name of the reference variable, put a dot, then type the method name, with the arguments in parentheses:

```
String s = new String("Help me!");
char c = s.charAt(3); //c gets 'p'
```

# Applied to other objects…

- It is exactly the same for non-**`String`** objects:

```
//instantiate turtle object
Turtle turtleOne = new Turtle(100,100,earth);
//move forward 100 pixels
turtleOne.forward();
//get current x position of turtleOne
int xPos = turtleOne.getXPos();
```

- Every kind of object has its own methods
- You will have to learn them (or look them up) if you want to use them