Tinker Academy

AP Computer Science Prep (Java DS & Algo)
Lecture 5 - Java Fundamentals 2
(Variables)

Winter 2015

Variables

Lecture 5 - Java Fundamentals 2

Variables

- Sometimes we need to keep track of a value
- Example, speed of a car
- The value could change

Lecture 5 - Java Fundamentals 2

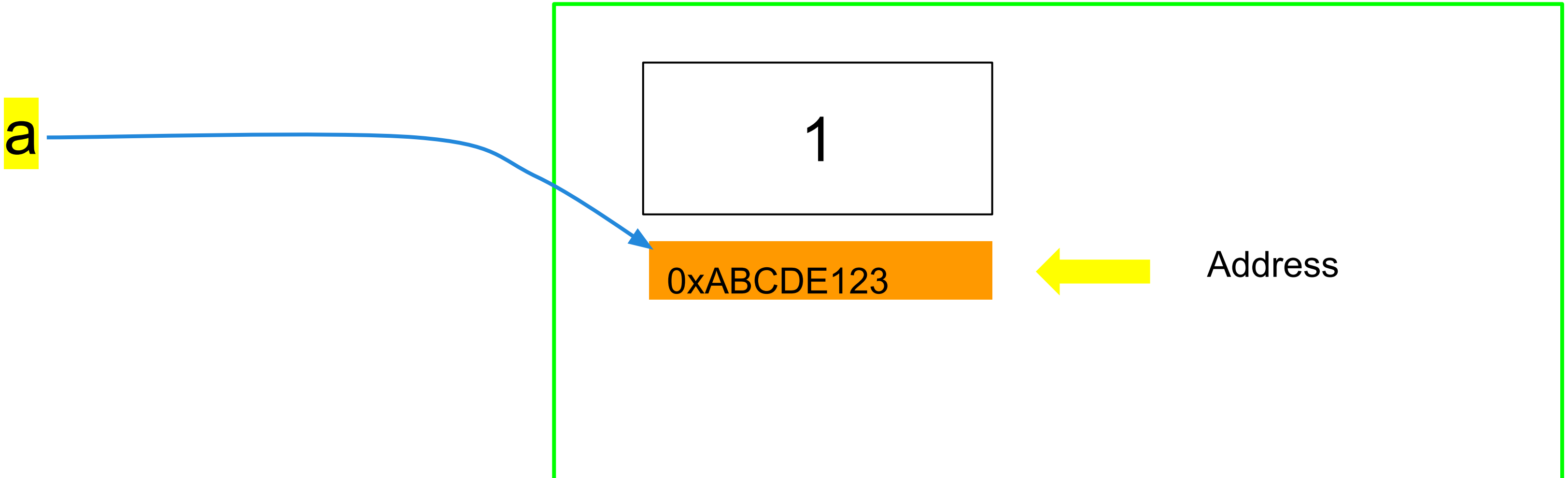
Variables

- Java has concept of a variable
- Used to keep track of a value that can vary

Lecture 5 - Java Fundamentals 2

Variables

- are actually named locations in memory
- a is a named location in memory
- address of the location is 0xABCDE123



Lecture 5 - Java Fundamentals 2

Variables

- are named storage locations
- point to the storage location in memory
- location can never be changed (unlike C/C++ pointers)
- value stored at location
- value can be changed
- always has a value

Lecture 5 - Java Fundamentals 2

Variables

- have a name
- name is case sensitive (case matters)
- name is a String (sequence of characters)
- first character of name should not be digit
- the value can be changed

Lecture 5 - Java Fundamentals 2

Variables

- Fields are variables
- Inputs to methods are variables
- Local variables are special variables created in a method

Declaring Variables

Lecture 5 - Java Fundamentals 2

Variables have associated data types

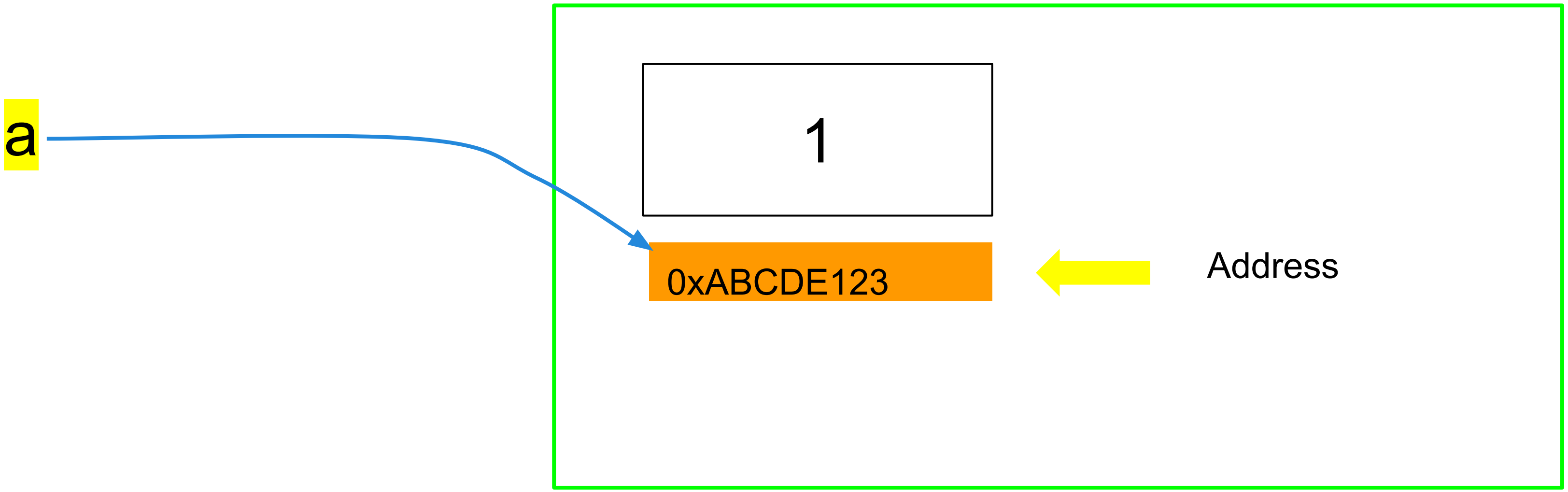
```
int a;
```

- Primitive DataType
- Reference DataType
- Special Datatype String

Lecture 5 - Java Fundamentals 2

Variables with Primitive DataType

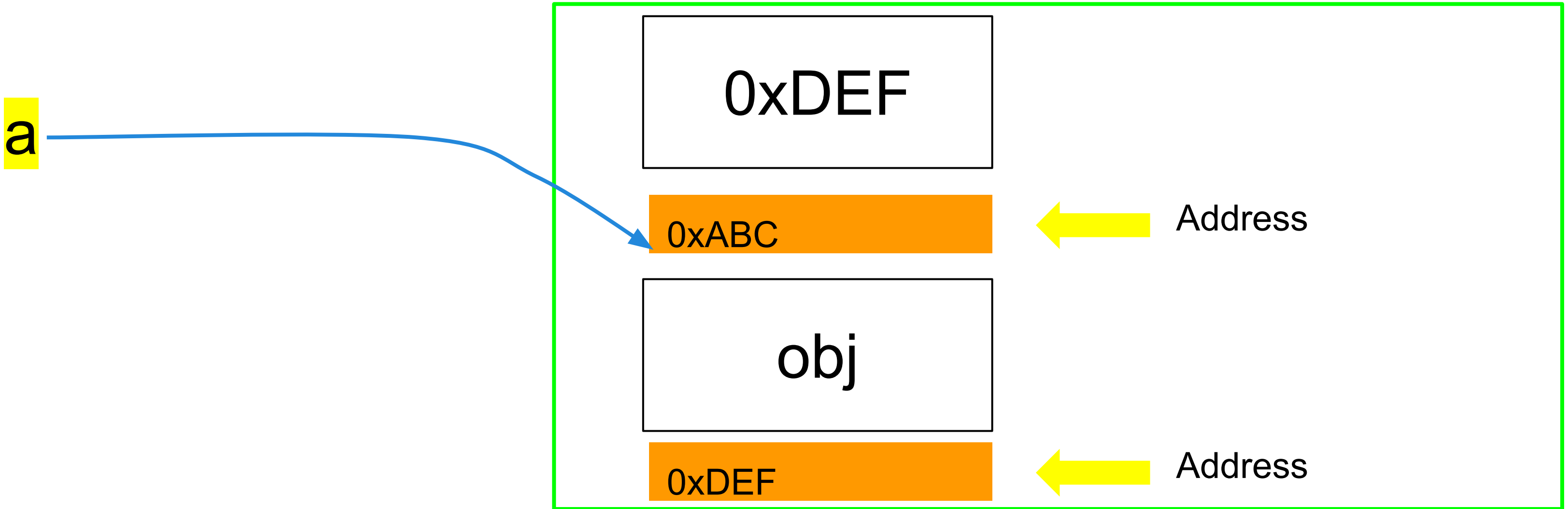
- value is a valid value for the data type



Lecture 5 - Java Fundamentals 2

Variables with Reference DataType

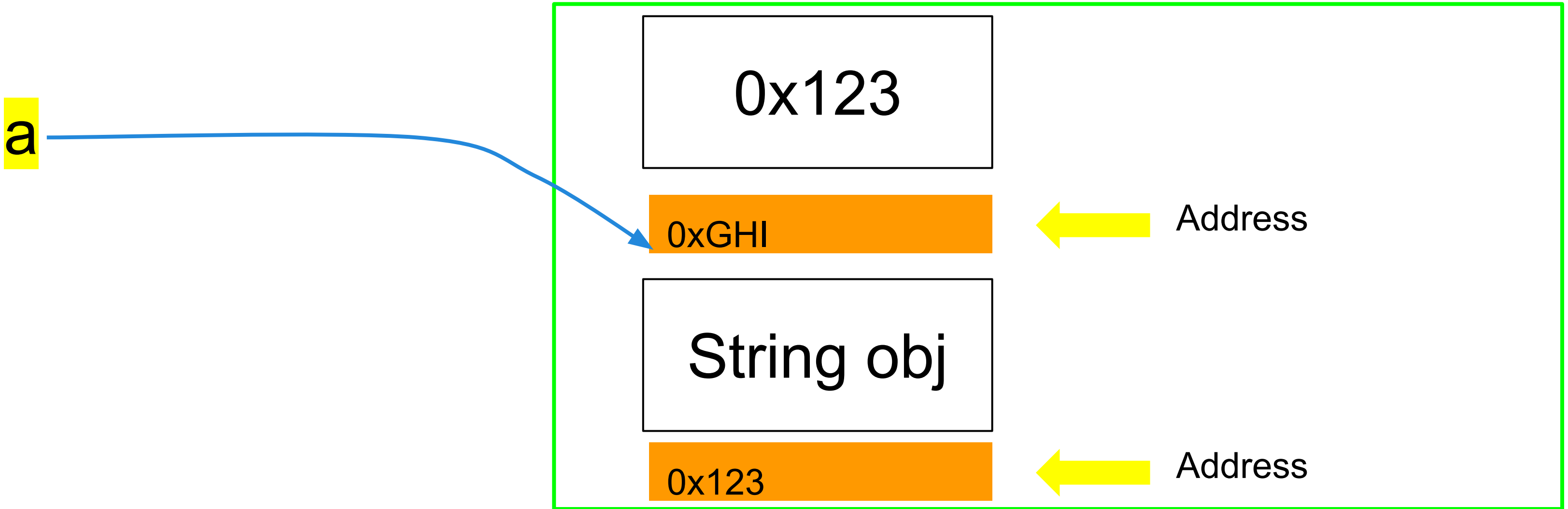
- value is a reference to an object OR
- a special value null (which indicates no reference)



Lecture 5 - Java Fundamentals 2

Variables with String DataType

- value is a reference to an String object OR
- a special value null (which indicates no reference)



Lecture 5 - Java Fundamentals 2

Variables always have an associated value

Variable	Name	Datatype	Value
int a;	a	int	0 (default value)

*Java Virtual Machine does not specify size of boolean, conceptually we are treating it here as 1 byte

Lecture 5 - Java Fundamentals 2

Declaring Variables

- variables need to be **declared** before use
- the datatype of the variable is specified during declaration
- the datatype of the variable can never change (Java is a statically typed language)

Lecture 5 - Java Fundamentals 2

Data Type Storage

Name	**Storage required**							
int	🟨	🟨	🟨	🟨				
long	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧
byte	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩
short	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦
float	🟪	🟪	🟪	🟪				
double	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪

Lecture 5 - Java Fundamentals 2

Data Type Storage

Name	Storage required							
char	🟦	🟦						
boolean*	🟥							
reference**								
String**								

*The Java language does not specify size of boolean explicitly - conceptually its 1 byte
** reference data types refer to objects and require varying size depending on the underlying class

Lecture 5 - Java Fundamentals 2

Declaring Variables

- the datatype indicates the amount of storage required
- if the initial value is not specified then variable gets a “default value”

Default Value

Lecture 5 - Java Fundamentals 2

Default Value

Variable	Datatype	Default Value
int a;	int	0
long a;	long	0
byte a;	byte	0
short a;	short	0
float a;	float	0
double a;	double	0

*Java Virtual Machine does not specify size of boolean, conceptually we are treating it here as 1 byte

Lecture 5 - Java Fundamentals 2

Default Value

Variable	**Datatype**	**Default Value**
char a;	char	0
boolean a;	boolean	False
Object a;	Object	null

*Java Virtual Machine does not specify size of boolean, conceptually we are treating it here as 1 byte

Initial Value

Lecture 5 - Java Fundamentals 2

Assigning an initial value

```
int a = 1;
```

*Java Virtual Machine does not specify size of boolean, conceptually we are treating it here as 1 byte

Lecture 5 - Java Fundamentals 2

Assigning an initial value

Variable	Datatype	Initial Value
int a = 1;	int	1
long a = 1L;	long	1
byte a = 1;	byte	1
short a = 1;	short	1
float a = 1.0f;	float	1.0
double a = 1.0;	double	1.0

*Java Virtual Machine does not specify size of boolean, conceptually we are treating it here as 1 byte

Lecture 5 - Java Fundamentals 2

Variables have associated data types

Variable	**Datatype**	**Initial Value**
char a = ‘C’;	char	‘C’ (number code
boolean a = True;	boolean	True
Object a = null;	Object	null
Object a = new Object();	Object	object of type Object
MyJavaClass a = new MyJavaClass();	MyJavaClass	object of type MyJavaClass

*Java Virtual Machine does not specify size of boolean, conceptually we are treating it here as 1 byte

Lecture 5 - Java Fundamentals 2

Initial Value

- initial value specified as part of declaration
- if the initial value is not specified then variable gets a "default value"
- Variables with Reference Datatypes can have a special value **null** which means the variable does not have a real reference (yet)

Assigning a new Value

Lecture 5 - Java Fundamentals 2

Variables have associated data types

Variable	Datatype	Initial Value	New Value
a = 2;	int	1	2
a = 2L;	long	1	2
a = 2;	byte	1	2
a = 2;	short	1	2
a = 2.0f;	float	1.0	2.0
a = 2.0;	double	1.0	2.0

*Initial Value was specified in an earlier side

Lecture 5 - Java Fundamentals 2

Variables have associated data types

Variable	Datatype	Initial Value	New Value
a = ‘D’;	char	‘C’ (number code 67)	‘D’ (number code 68)
a = False;	boolean	True	False
a = null;	Object	null	null
a = new Object();	Object	object of type Object	another object of type Object*
a = new MyJavaClass();	MyJavaClass	object of type MyJavaClass	another object of type MyJavaClass*

*the old objects are eventually destroyed by the JVM

Lecture 5 - Java Fundamentals 2

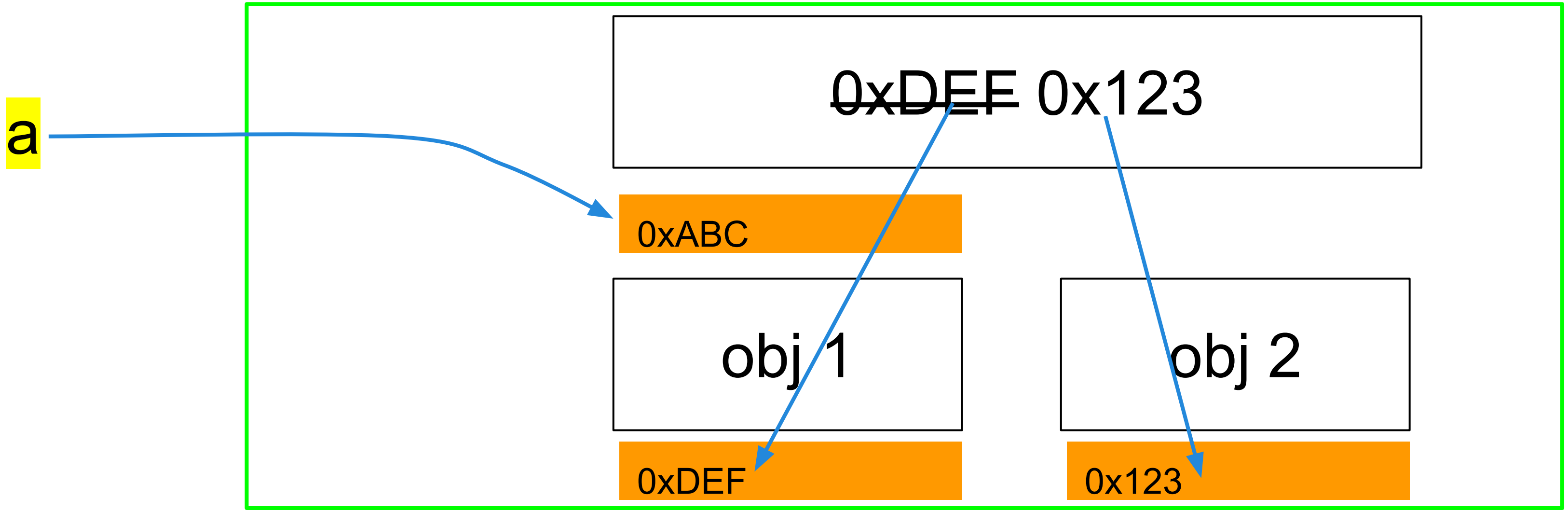
Assigning a new Value

- new value specified by using an assignment statement
- new value can be the special value **null** indicating that the variable no longer points to a object reference
- new value gets replaces the old value

Lecture 5 - Java Fundamentals 2

Assigning a new Value

- new reference value replaces old reference value



Lecture 5 - Java Fundamentals 2

Assigning a new Value

- new value specified by using an assignment statement
- new value can be the special value **null** indicating that the variable no longer points to a object reference
- new value gets replaces the old value
- **if** the old value is an object reference, **and** the object has no other references to it, the object is eventually gets destroyed by the JVM