Using Objects from the API

DEBORAH A. TRYTTEN

CS 1324

Classes and Objects: What we know so far

- Classes with a main program
  - Most programs we've written
- Classes with a main program and static methods
- These are not typical classes
  - They can't be used to build objects
  - Most of the classes in the Java API can be used to build objects
  - Exceptions we know: Math, Arrays classes

Objects

- Most objects conceal
  - Some data
    - Can be primitive data types or other objects
  - Data represents the properties of the object
  - Usually cannot access data directly
- Most objects provide
  - Constructors to initialize the data
  - Methods to access the concealed data
    - Accessors
  - Methods to change the concealed data
    - Mutators

Example: String objects

- Create a reference

```
String name;
```

- Construct the object

```
name = new String(“Raven”);
name = “Jazz”;  // hidden constructor
```

- Manipulate the object using methods that are not static

```
String initial = name.substring(0,1);
if (initial.equals(“R”)) System.out.println(“Raven?”);
String smallName = name.toLowerCase();
```

Example: Scanner

- Create a reference

```
Scanner keyboard;
```

- Construct an object

```
keyboard = new Scanner(System.in);
```

- Manipulate the object with methods that are not static

```
for (int i=0; i<array.length && (keyboard.hasNext()); ++i)
{
 array[i] = keyboard.next();
}
```

Syntax Summary

- Declare a reference
  ClassName objectReference;
- Construct an object (initialization)
  - objectReference = new ClassName(argument(s));
- Call non-static methods using dot notation
  - objectReference.methodName(argument(s))
  - The objectReference is like an argument—provides input to the method
- Call static methods using dot notation
  - ClassName.methodName(argument(s))

Static versus Non-static Methods

- When a class is first used in a program, it is brought into memory
  - Remains in memory
  - It is static
- When objects are constructed, they remain in memory only as long as they are referenced
  - The are dynamic (i.e. not static)
- The keyword static means that the method (or data) is owned by the class instead of being owned by an individual object
- Example: public static void main(String[] args)

Examine A New Class

- https://docs.oracle.com/en/java/javase/13/docs/api/java.desktop/java/awt/Point.html
- Point class
  - Used to store ordered pairs (like coordinates)
- Data
  - Unusual for data to be public
- Constructors
  - Used to initialize Point objects
- Accessor methods
- Mutator methods
- What method names indicate accessor vs mutator?

Example

- Create three Point objects that are located at the origin
  - Uses constructors
- Write out the data in a Point object as {x,y}
  - Use accessors
- Find three ways to place one of the Point objects created to (3, 5)
  - Use mutators
- Show memory diagram
- Write a code fragment that determines whether two Point objects are in the same location
  - Accessor or mutator?

Instant Quiz Question 1

- How many objects are constructed in the code below?

```
Point point1, point2;
point1 = new Point (1, 3);
point2 = new Point();
point2 = new Point (point1);
```

- a) 1 b) 2 c) 3 d) 4

Another Class: Random

- https://docs.oracle.com/en/java/javase/13/docs/api/java.base/java/util/Random.html
- Write a code fragment that prints out 100 random numbers between 1 and 100 using this class
  - We can use the class even though we have no idea how the Random numbers are created
  - We don't even really know whether the methods are accessors or mutators (they are both!)

Instant Quiz Question 2

- The Random class has a non-static method described below;

boolean nextBoolean(): Generates the next boolean value from this random number generator's sequence.

If there is a Random reference randBool declared and constructed,

which syntax is the correct way to call this method?

a) Random.nextBoolean(false);

b) Random.nextBoolean();

c) randBool.nextBoolean(true);

d) randBool.nextBoolean();

Wrapper Classes

- Java has a class related to each primitive data type
  - Double
  - Integer
  - Character
- Are objects in these classes mutable or immutable?
  - https://docs.oracle.com/en/java/javase/13/docs/api/java.base/java/lang/Integer.html
- Why do these classes exist?
  - We can do some really cool programming with objects that can't be done with primitive data types
  - To organize useful methods (similar to Math/Arrays)

Examples

- Read a String containing a floating point number and create a double without using Scanner
  - Convert it to a double if it is a finite number
- Compare: use a static method in the Double class to convert a String to a double

- Note: Be careful with Double versus double now

Think, Pair, Share

- Write a method below that returns true if a String contains only digits (‘0’ to ‘9’)
- Use the isDigit() method in the Character class
- https://docs.oracle.com/en/java/javase/13/docs/api/java.base/java/lang/Character.html

```
public static boolean isAllDigits(String source)
```

String versus StringBuilder

- Java has two classes that store sequences of characters
  - String (immutable)
    - Perfect size char[]
  - StringBuilder (mutable)
    - Oversize char[]
- String operations like + actually use StringBuilder behind the scenes
- We were able to use String and StringBuilder objects before we knew about arrays
  - Encapsulation
  - One of the advantages of objects/classes

Design Decisions

- By making String immutable, Java is able to be incredibly efficient in how it stores String objects
  - This matters because there are so many String objects
- Example:
  - StringBuilder objects have to have extra space for characters that can be added later
  - String objects do not need any extra space
- Having multiple classes allows Java programmers to make good design decisions

Instant Quiz Question 3

- Suppose we were storing the name of the business in a mall which a person was closest too at any given time. Which class should be used to represent the businesses?

a) String

b) StringBuilder

c) Integer

d) Character

What is a Class?

- Classes in the Java API are bigger and more complicated data types
  - Data types contain data and legal operations
- int
  - Stores an integral number in binary
  - Has +, -, *, / provided by compiler
- String
  - Stores a sequence of characters
  - Has methods provided by Java API
  - One String class, lots of String objects

Classes

- A prototype for objects
- A contract for the state of objects
  - State means the data and the behavior
  - Each object from the class will have specific data stored
    - Example: Every String has a perfect size char[]
  - Each object has specific behaviors
    - Example: String objects are immutable
  - Methods that change the state will result in another legal state
    - Example: The toLowerCase method in the String class has to return a String because Strings are immutable

What is an Object?

- A specimen of its class
- Multiple objects from the same class will have the same data elements/types of data and behaviors
  - May have different data stored
- Classes are like cookie cutters
  - Objects are like cookies
    - Each cookie can have its own decorations

Classes versus Objects

- Subtle but important difference
- Goal: Find a class with objects that you are really comfortable with
  - Use your knowledge and understanding of this class to generalize your knowledge and understanding of objects and classes
  - Helps if the class has both static and non-static methods
- String not a great choice
  - Immutable
  - Crazy syntax (like constructors that don't use new and +)
- ArrayList class is great choice