

## Chapter 4 Review

Tuesday, December 18, 2012  
7:58 AM

1. Consider the following class declaration.

```
public class IntCell {  
    private int myStoredValue;  
  
    // constructor not shown  
  
    public int getValue() {  
        return myStoredValue;  
    }  
  
    public String toString() {  
        return "" + myStoredValue;  
    }  
}
```

Assume that the following declaration appears in a client class.

```
IntCell m = new IntCell();
```

Which of these statements can be used in the client class?

I. `System.out.println(m.getValue());`

II. `System.out.println(m.myStoredValue);`

III. `System.out.println(m);` *.toString() gets called*

- a) I only
- b) II only
- c) III only
- d) I and II
- e) I and III

2. Consider the following static method.

```
public static int calculate(int x) {  
    ✓ x = 4 + 4;  
    ✓ x = 3 + 3;  
    ✓ x = 1/2 + 1/2;  
    return x;  
}
```

$$\begin{array}{r} x \\ 4 \\ \hline 4 \\ 8 \\ 6 \\ 32 \end{array}$$

Which of the following can be used to replace the body of `calculate` so that the modified version of `calculate` will return the same result as the original version for all `x`?

- a) `return 3 + x;`
- b) `return 3 * x;`
- c) `return 4 * x;`
- d) `return 6 * x;`
- e) `return 8 * x;`

4. Consider the following class declaration:

```
public class Person {  
    private String firstName;  
    private String lastName;  
    private int age;  
  
    public Person(String fn, String ln, int a) {  
        firstName = fn;  
        lastName = ln;  
        age = a;  
    }  
  
    public String getFirstName() {  
        return firstName;  
    }  
  
    public String getLastName() {  
        return lastName;  
    }  
  
    public int getAge() {  
        return age;  
    }  
}
```

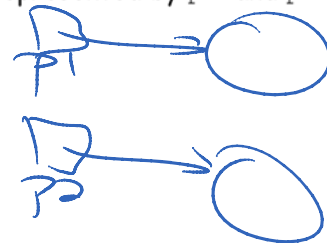
Assume that variables `p1` and `p2` have been declared as follows:

Clint

`Person p1, p2;`

Which of the following is the best way to test whether the people represented by `p1` and `p2` have the same first name?

- a) `p1 == p2`
  - b) `p1.getFirstName().equals(p2.getFirstName())`
  - c) `p1.getFirstName() == p2.getFirstName()`
  - d) `p1.equals(p2)`
  - e) `p2.equals(p1)`
- Handwritten notes:   
 - An arrow points from "string" to option b.   
 - An arrow points from "string" to option c.   
 - A large arrow points from option c to the text "Can't use == to compare strings".



4. Assume that a class includes the following three methods:

```
public static int min(int x, int y) {  
    if (x < y) return x;  
    else return y;  
}  
  
public static int min(String s, String t) {  
    if (s.length() < t.length()) return s.length();  
    else return t.length();  
}  
  
public static void testMin() {  
    System.out.println(min(3, "Hello"));  
}
```

int, String

Which of the following best describes what happens when this code is compiled and executed?

- a) The code will not compile because the types of the arguments used in the call to `min` do not match the types of the parameters in either version of `min`.
- b) The code will not compile because it includes two methods with the same name and the same return type.
- c) The code will not compile because it includes two methods with the same name and the same number of parameters.
- d) The code will compile and execute without error; the output will be 3.
- e) The code will compile and execute without error; the output will be 5.

5. Consider the following method. *SomeCode is required to return an int.*

```
public int someCode(int a, int b, int c) {
    if ((a < b) && (b < c)) {
        return a;
    }
    if ((a >= b) && (b >= c)) {
        return b;
    }
    if ((a == b) || (a == c) || (b == c)) {
        return c;
    }
}
```

*if all these are false, it doesn't know what to return.*

Which of the following best describes why this method does not compile?

- a) The reserved word `return` cannot be used in the body of an `if` statement.
- ☒ b) It is possible to reach the end of the method without returning a value.
- c) The `if` statements must have `else` parts when they contain `return` statements.
- d) Methods cannot have multiple `return` statements.
- e) The third `if` statement is not reachable.

The next two questions refer to the `Point` class and the `Circle` class declarations (attached).

6. In a client program which of the following correctly declares and initializes `Circle` `circ` with center at (29.5, 33.0) and radius 10.0?

- ☒ a) `Circle circ = new Circle(29.5, 33.0, 10.0);`
- ☒ b) `Circle circ = new Circle((29.5, 33.0), 10.0);` *double double double*
- ☒ c) `Circle circ = new Circle(new Point(29.5, 33.0), 10.0);` *Point double*
- ☒ d) `Circle circ = new Circle();`  
`circ.myCenter = new Point(29.5, 33.0);`  
`circ.myRadius = 10.0;`
- ☒ e) `Circle circ = new Circle();`  
`circ.myCenter = new Point();`  
`circ.myCenter.myX = 29.5;`  
`circ.myCenter.myY = 33.0;`  
`circ.myRadius = 10.0;` *these are private* *private*

7. Which of the following would be the best specification for a `Circle` method `isInside` that determines whether some `Point` lies inside the `Circle`?

- a) `public boolean isInside()`
- b) `public void isInside(boolean found)`
- ☒ c) `public boolean isInside(Point p)`
- d) `public void isInside(Point p, boolean found)`
- e) `public boolean isInside(Point p, Point center, double radius)`

The next three questions refer to the Time class declaration (attached).

7. Which of the following is a *false* statement about the methods of Time?

- ☒ a) equals, lessThan, and toString are all accessor methods.
- ☒ b) increment is a mutator method.
- ☒ c) Time() is the default constructor.
- ☒ d) The Time class has three constructors.
- ☒ e) There are not static method in this class.

8. Which of the following represents correct *implementation code* for the constructor with parameters?

- a) myHrs = 0;  
myMins = 0;  
mySecs = 0;
- ☒ b) myHrs = h;  
myMins = m;  
mySecs = s;
- c) resetTime(myHrs, myMins, mySecs);
- d) h = myHrs;  
m = myMins;  
s = mySecs;
- e) Time = new Time(h, m, s);

9. A client program has a display method that writes the time represented by its parameter. Note again that this method is in a client program and not the Time class:

```
// output time t in the form hrs:mins:secs  
public static void display(Time t) {  
    /* implementation not shown */  
}
```

Which of the following are correct replacements for the implementation code?

- ☒ I. ~~Time t = new Time(h, m, s);~~ System.out.println(T);  
don't make another time.  
we don't have h, m, s
- ☒ II. System.out.println(t.myHrs + ":" + t.myMins + ":" + t.mySecs);
- ☒ III. System.out.println(t);  
→ print t  
→ print t  
→ print t  
→ call toString()

- a) I only
- b) II only
- ☒ c) III only
- d) II and III only
- e) I, II, and III

The next five questions refer to the following `Date` class declaration (attached).

1. Which of the following correctly constructs a `Date` object?

- a) `Date d = new (2, 13, 1947);` → *Date*
- ☒ b) `Date d = new Date(2, 13, 1947)`
- c) `Date d;`  
`d = new (2, 13, 1947);` → *Date*
- d) `Date d;`  
`d = Date(2, 13, 1947);` → *new*
- e) `Date d = Date(2, 13, 1947);` → *new*

2. Which of the following will cause an error message?

*Good* I. `Date d1 = new Date(8, 2, 1947);`  
`Date d2 = d1;`

*Good* II. `Date d1 = null;`  
`Date d2 = d1;`

☒ III. `Date d = null;`  
`int x = d.year();`

- a) I only
- b) II only
- ☒ c) III only
- d) II and III only
- e) I, II, and III

*you can't call a method on a null reference object*

*You don't need to know #12*

3. A client program creates a `Date` object as follows:

```
Date d = new Date(1, 13, 2002);
```

Which of the following subsequent code segments will cause an error?

- Good* a) `String s = d.toString();`
- Good* b) `int x = d.day();`
- Good* c) `Date e = d;`
- Good* d) `Date e = new Date(1, 13, 2002);`
- ☒ e) `int y = d.myYear;`

*→ private*

4. Consider the implementation of a `write()` method that is added to the `Date` class:

```
// Write the date in the form m/d/y, for example 2/17/1948
public void write() {
    /* implementation not shown */
}
```

Which of the following could be used for the implementation code?

- I. `System.out.println(myMonth + "/" + myDay + "/" + myYear);`
- II. `System.out.println(month() + "/" + day() + "/" + year());`
- III. `System.out.println(this);`

- a) I only
- b) II only
- c) III only
- d) II and III only
- e) I, II, and III

Don't need to know "this"  
It refers to the current object

5. Here is a short program that uses Date objects.

```
public class BirthdayStuff {  
    public static void main(String[] args) {  
        Date d = findBirthdate();  
    }  
  
    public static Date findBirthdate() {  
        Date bd;  
        /* code missing here to get bd */  
        return bd;  
    }  
}
```

*bd = new Date( \_\_, \_\_, \_\_ )*

Which of the following is a correct replacement for /\* code missing here to get bd \*/?

☒ I. `System.out.println("Enter birthdate: mo, day, yr: ");`  
`int m = Keyboard.readInt();`  
`int d = Keyboard.readInt();`  
`int y = Keyboard.readInt();`  
`bd = new Date(m, d, y);`

☒ II. `System.out.println("Enter birthdate: mo, day, yr: ");`  
`int bd.month() = Keyboard.readInt();`  
`int bd.day() = Keyboard.readInt();`  
`int bd.year() = Keyboard.readInt();`  
`bd = new Date(bd.month(), bd.day(), bd.year());`

*Doesn't make sense*

☒ III. `System.out.println("Enter birthdate: mo, day, yr: ");`  
`int bd.myMonth() = Keyboard.readInt();`  
`int bd.myDay() = Keyboard.readInt();`  
`int bd.myYear() = Keyboard.readInt();`  
`bd = new Date(bd.myMonth(), bd.myDay(), bd.myYear());`

*Doesn't make sense*

- ☒ a) I only
- b) II only
- c) III only
- d) I and II only
- e) I and III only



```

private double myX;
private double myY;

// postcondition: this Point has coordinates (0,0)
public Point() {
    /* implementation not shown */
}

// postcondition: this Point has coordinates (x,y)
public Point(double x, double y) {
    /* implementation not shown */
}

// other methods not shown

public class Circle {
    private Point myCenter;
    private double myRadius;

    // postcondition: this Circle has center at (0, 0) and radius 0.0
    public Circle() {
        /* implementation not shown */
    }

    // postcondition: this Circle has the given center and radius
    public Circle(Point center, double radius) {
        /* implementation not shown */
    }

    // other methods not shown

    public boolean isInside ( Point p ) {
}

```

Handwritten annotations: A bracket on the left of the `Circle` class definition. In the `Circle` constructor, `Circle()` is circled. In the `Circle(Point center, double radius)` constructor, the entire signature is circled. Two arrows point from the circled signature to the words "Point" and "double" written below it. The word "boolean" is written above "isInside" in the `isInside` method signature.

```
-private int myHrs;  
-private int myMins;  
-private int mySecs;
```

```
const. public Time() {  
    /* implementation not shown */  
}
```

```
const. public Time(int h, int m, int s) {  
    /* implementation not shown */  
}
```

```
// resets time to myHrs = h, myMins = m, mySecs = s
```

```
public void resetTime(int h, int m, int s) { → mutator  
    /* implementation not shown */  
}
```

```
// advances time by one second
```

```
public void increment() { → mutator  
    /* implementation not shown */  
}
```

```
// returns true if this time equals t, false otherwise
```

```
public boolean equals(Time t) { → accessor  
    /* implementation not shown */  
}
```

typo

```
// resets time to myHrs = h, myMins = m, mySecs = s.
```

```
public boolean lessThan(Time t) { → accessor  
    /* implementation not shown */  
}
```

```
// returns tims as a SString in the form hrs:mins:secs
```

```
public String toString() { → accessor  
    /* implementation not shown */  
}
```

```

private int myDay;
private int myMonth;
private int myYear;

public Date() {
    /* implementation not shown */
}

public Date(int mo, int day, int yr) {
    /* implementation not shown */
}

public int month() {    // returns the month of Date
    /* implementation not shown */
}

public int day() {      // returns the day of Date
    /* implementation not shown */
}

public int year() {     // returns the year of Date
    /* implementation not shown */
}

// string representation fo Date as "m/d/y, e.g. 4/18/1985
public String toString() {
    /* implementation not shown */
}

```

*public void write() {*  
*so, pp (toString)*