CS 61B Fall 2020 — Topical Review Session, Section 2: Inheritance, Static & Dynamic, DMS

MT1 Review

1 Static vs Non-static Practice

(a) Fill in the comment sin the main method to indicate what is outputted by each print line command. If there is an error, write CE in the blank to indicate a compile-time error, and RE to indicate runtime error.

```
public class Cereal {
    int numMarshmallows;
    static double crunchiness;

    public Cereal() {
        numMarshmallows = 100;
        crunchiness = 1.0;
    }

    static void pour() {
        crunchiness -= 0.1;
    }

    void eat (int quantitity) {
        numMarshmallows -= quanitty;
        if(quantity > 10) {
            crunchiness -= 0.05;
        }
    }
}
```

continued on the next page...

```
public static void main(String[] args) {
    Cereal a = new Cereal();
    a.pour();

    System.out.println(a.numMarshmallows);   100
    System.out.println(a.crunchiness);   0.9
    System.out.println(Cereal.crunchiness);   0.9

    Cereal b = new Cereal();
    a.pour();

    System.out.println(a.crunchiness);   0.9
    System.out.println(b.crunchiness);   0.9

    b.eat(5);

    System.out.println(a.numMarshmallows);   100
    System.out.println(b.numMarshmallows);   95

    a.eat(15);

    System.out.println(a.numMarshmallows);   85
    System.out.println(b.numMarshmallows);   95
    System.out.println(a.crunchiness);   0.85
    System.out.println(b.crunchiness);  0.85
}
```

(b) Suppose we add the following method to our Cereal class and call it from the main method as shown:

```
public class Cereal {
    ...
    static void devour() {
        numMarshmallows = 0;
    }
    ...
    public static void main(String[] args) {
     ...
        Cereal.devour();
    }
}
```

What will happen when we run the code?

Solution: We would get a compiler error. numMarshmallows is not a static variable.

2 Dynamic Method Selection with Casting

Suppose we have the following Dog, Corgi, and Retriever classes:

```
public class Dog {
    public void bark() {}
}

public class Corgi extends Dog {
    public void herd() {}
}

public class Retriever extends Dog {
    public void swim {}
}
```

For each line below, write CE if there is a compiler error, RE if there is a runtime error, or nothing if there are no errors.

```
public static void main(String[] args) {
    Dog dog = new Dog();
    Corgi corgi = new Corgi();
    Dog bob = new Corgi();

    ((Dog) corgi).bark();   nothing
    ((Dog) corgi).herd();   CE
    ((Corgi) corgi).herd();   nothing
    ((Corgi) dog).bark();     RE
    ((Corgi) dog).herd();     RE
    ((Retriever) corgi).swim()    CE
}
```

3 Static vs Dynamic Types

```
public class Fingerprint {...}
public class Key { ... }
public class SkeletonKey extends Key { ... }

public class StandardBox { public void unlock(Key k) { ... } } // UK

public class BioBox extends StandardBox {
    public void unlock(SkeletonKey sk) { ... } // USK
    public void unlock(Fingerprint f) { ... } // UF
}
```

For each of the lines below, indicate what the output would be (UK, USK, or UF). If there will be a compile-timer error, write CE and if there will be a run-time error, write RE.

```
public static void doStuff(Key k, SkeletonKey sk, Fingerprint f) {
    StandardBox sb = new StandardBox();
    StandardBox sbbb = new BioBox();
    BioBox bb = new BioBox();

    sb.unlock(k);   UK
    sbbb.unlock(k); UK
    bb.unlock(k);   UK

    sb.unlock(sk);  UK
    sbbb.unlock(sk);  UK
    bb.unlock(sk);  USK

    sb.unlock(f);   CE
    sbbb.unlock(f);  CE
    bb.unlock(f);   UF

    bb = (BioBox) sbbb; No error

    ((StandardBox) bb).unlock(sk);      UK
    ((StandardBox) sbbb).unlock(sk);    UK
    ((BioBox) sb).unlock(sk);           RE
}
```