Things You Should Know for Test 2

1. How to refactor procedural code into functions and classes.

2. How to trace programs containing functions and objects.

3. How to identify: signatures, constructors, head, body, definiton, member variables, member functions, method calls, parameters, and variable scopes.

4. What the difference between public and private is.

5. What the difference is between static functions of a class and methods of an object.

6. How to design your own classes.

7. How to overload a function.

8. How to override a function.

9. What the difference between pass by value and pass by reference is.

10. How to write a toString method and when that method is invoked by the compiler.

11. How to write setters and getters.

12. What encapsulation is and why we would organize programs into classes and objects.

13. What the scope of a variable is and how long various kinds of variables last for.

14. What the advantages of an Object-oriented program is over a procedural one.

15. How to use an existing object-oriented API.

Test 2: Functions, Classes, and Objects

Name_____

```
public class Armor{

  public Armor(Hero h){
    this.owner = h;
    this.shieldModifier = 1;
    this.name = "none";
  }

  public Armor(Hero h, int mod){
    this.owner = h;
    this.shieldModifier = mod;
  }

  public int shield(Monster m){
    return m.strength / shieldModifier;
  }

  public void equip(Hero h){
    owner = h;
  }

  public void unequip(){
    owner = null;
  }

  public boolean hasOwner(){
    return owner == null ? false : true;
  }

  private String name;
  private Hero owner;
  private int shieldModifier;
}



public class Weapon {

  public Weapon(Hero o){
    this.name = "bare hands";
    this.strengthModifier = 0;
    this.owner = o;
  }

  public Weapon(String name, int mod){
    this.name = name;
    this.strengthModifier = mod;
  }

  public String getName(){
    return name;
  }

  public int getStrengthModifier(){
    return strengthModifier;
  }


  public int use(Monster m){
    return
m.takeDamage(owner.getStrength() +
strengthModifier);
  }


  public void equip(Hero h){
    this.owner = h;
  }

  public Weapon unequip(){
    this.owner = null;
  }

  public boolean hasOwner(){
    return owner == null ? false : true;
  }

  private String name;
  private int strengthModifier;
  private Hero owner;
}
```

```
public class Monster{

  public Monster(String name, int maxHealth, int
strength){
    this.name = name;
    this.maxHealth = maxHealth;
    this.maxStrength = strength;
    this.health = health;
    this.strength = strength;
  }

  public int getHealth(){
    return this.health;
  }

  public int getStrength(){
    return this.strength;
  }

  public int getMaxHealth(){
    return this.maxHealth;
  }

  public int getMaxStrength(){
    return this.maxStrength;
  }

  public int attack(Hero h){
    return h.takeDamage(this.getStrength());
  }

  public int takeDamage(int amount){
    health -= amount;

    int damageAmount = amount;

    if(health < 0){
  damageAmount += health;
  health = 0;
    }

    return damageAmount;
  }

  public boolean isDead(){
    return health <= 0 ? true : false;
  }

  public int weaken(int value){
    this.strength -= value;

    int amountWeakend = value;

    if(strength < 0){
      amountWeakened += strength;
      strength = 0;
    }

    return amountWeakened;
  }

  public int strengthen(int value){
    this.strength += value;

    return value;
  }

  public int heal(int amount){
    health += amount;
    int amountHealed = amount;

    if(health > maxHealth){
      amountHealed -= (health - maxHealth);
      health = maxHealth;
    }

    return amountHealed;
  }

  private String name;
  private int maxHealth;
  private int health;
  private int strength;
  private int maxStrength;
}
```

```
public class Hero{

  public Hero(String name, int maxHealth, int
strength){
    this.name = name;
    this.maxHealth = maxHealth;
    this.health = maxHealth;
    this.strength = strength;
    this.maxStrength = strength;
    this.armor = new Armor(this);
    this.weapon = new Weapon(this);
    this.spell = new Spell(this);
  }

  public getName(){
    return this.name;
  }

  public getHealth(){
    return health;
  }

  public getStrength(){
    return strength;
  }

  public int getMaxHealth(){
    return maxHealth;
  }

  public int getMaxStrength(){
    return maxStrength;
  }

  public equipWeapon(Weapon w){
    this.dropWeapon();
    this.weapon = w.equip();
  }

  public equipArmor(Armor a){
    this.dropArmor();
    this.armor = a.equip();
  }

  public Weapon dropWeapon(){
    Weapon w = this.weapon;

    w.unEquip();
    this.weapon = new Weapon();

    return w;
  }

  public Armor dropArmor(){
    Armor a = this.armor;
    a.unEquip();
    this.armor = new Armor();

    return a;
  }

  public grantSpell(Spell s){
    this.spell = s;
  }

  public int castSpell(Monster m){
    return this.spell.cast(m);
  }

  public int castSpell(Hero h){
    return this.spell.cast(h);
  }

  public int attack(Monster m){
    return this.weapon.use(m);
  }

  public int takeDamage(int damage){
    return assignDamaged(armor.shield(damage));
  }

  private int assignDamage(int amount){
    int amountDamaged = amount;
    health -= amount;

    if(health < 0){
      amountDamaged -= (amount + health);
      health = 0;
    }

    return amountDamaged;
  }

  public boolean isDead(){
    return health <= 0 ? true : false;
  }

  public int heal(int amount){

    int amountAdded = amount;
    health += amount;


    if(health > maxHealth){
      amountAdded = amount - (health - maxHealth);
      health = maxHealth;
    }

    return amountAdded;
  }

  private String name;
  private int maxHealth;
  private int health;
  private int strength;
  private Spell spell;
  private Weapon weapon;
  private Armor armor;
  private int maxStrength;
}
```

Recognize Elements

Use the program on page 2 to answer the questions in this section.

1. How many classes are in the program listed?

2. How many constructors does the Armor class have?

3. How many methods are in the Monster class?

4. What is the signature of the use() method in the Weapon class?

5. Name the private member variables of the Hero class.

6. What is the return type of the dropWeapon() method in the Hero class?

7. On what classes does the Hero class depend?

8. Name the parameters of the Monster constructor.

9. What does the method attack() of the Hero class do?

Trace Programs

```
public class RussianDoll{

  public RussianDoll(String name){
    this.name = name;
  }


  public RussianDoll(String name, RussianDoll r){
    this.name = name;
    this.r = r;
  }


  public RussianDoll getDoll(){
    return r;
  }


  public String getName(){
    return name;
  }


  public void insertDoll(RussianDoll r){
    this.r = r;
  }


  public String toString(){
    return this.name;
  }


  private String name;
  private RussianDoll r;

}
```

```
public class DollChain {

  public DollChain(RussianDoll r){
    this.outermost = r;
  }


  public DollChain(String name){
    this.outermost = new RussianDoll(name);
  }


  public void add(RussianDoll r){
    r.insertDoll(outermost);
    outermost = r;
  }

  public RussianDoll removeOutermost(){
    RussianDoll previousOutermost = this.outermost;

    this.outermost = this.outermost.getDoll();

    return previousOutermost;
  }

  public toString(){
    RussianDoll t = outermost;
    String s = “”;

    s += t.toString();

    while(t.getDoll() != null){
      t = t.getDoll();
      s += t.toString();
      s += “ “;
    }

    return s;
  }



  private RussianDoll outermost;
}
```

1.

```
RussianDoll first = new RussianDoll(“Bob”);
RussianDoll second = new RussianDoll(“Sally”);
RussianDoll third = new RussianDoll(“Tamara”);

DollChain c = new DollChain(first);
c.add(second);
c.add(third);

System.out.println(c);
```

2.

```
RussianDoll a = new RussianDoll(“a”);
RussianDoll b = new RussianDoll(“b”);

a.insertDoll(b);
b.insertDoll(a);

System.out.println(a.getDoll().getDoll());
System.out.println(a.getDoll());
System.out.println(a);
```

Use an Existing API

Shape Factory, RPG, Dating Simulator

```
public class Person{

  public Person(String name){
    population++;
    this.name = name;
  }


  public boolean isZombie(){
    return isZombie;
  }


  public static int getPopulation(){
    return population;
  }


  public String getName(){
    return name;
  }


  public void becomeZombie(){
    this.isZombie = true;
    population--;
    zombieCount++;
  }


  public void die(){
    if(!isZombie){
      population--;
    }
  }


  public void bite(Person p){
    if(this.isZombie){
      p.becomeZombie();
    }
  }


  private boolean isZombie;
  private static int population;
  private static int zombieCount;
  private String name;
}
```

Using the Person class, create 6 Person objects. Print out the population. Make one of the Person object become a zombie. Have that person who became a zombie bite one of the other people. Print out the population. Have one person who is not a zombie bite another person who is not a zombie. Print out the population.

Questions

1. What is encapsulation?

2. What is the difference between pass by value and pass by reference?

3. What is a constructor?

4. What is a function signature?

5. What is the difference between a static method and an ordinary method?

6. Why is it a best practice to make our member variables of a class private?

7. What is the difference between a public and private method?

8. What is the signature of a function?

9. What is the difference between a function's definition and a function's invocation or application (when you call it with the parentheses such as Math.pow(3,2))?

10. When would you want to overload a function?

11. What is the return type of a function?

12. When you see the keyword new, what function is being called at this point?

13. What is the difference between a class definition and an instantiated object of the class?

14. What is the scope of a variable?

15. How long does a parameter passed into a function exist for?

16. What is the difference between an Object type and a primitive type?

17. What is the difference between the way objects and primitive variables handled when they are passed as parameters to a function?

18. How many files can be in a project?

19. How many packages can be in a project?

20. How many files can be in a package?

21. How many classes can be in a file?

22. Under what circumstance is the toString() method implicitly called on an object?

23. When do you use the return type void?

Design an Object-Oriented Program

Write an Ecosystem program. Besides the Ecosystem simulation class which is where your main function occurs, you have 2 other classes: a Deer and a Lion class. The Deer and Lion classes each have a private member variable for a name (to uniquely identify each object), an x coordinate, and a y coordinate (for the location of the object in the Ecosystem). The constructor for each of these classes takes an x, y, and name parameter as input to initialize the private member variables. The Lion and Deer classes also each have a void move(direction) method. When an animal moves, it increments or decrements its x or y coordinate depending on which direction it moved (north, south, east, or west). Additionally, the Lion class has a void eat(Deer) method. The Ecosystem class is in charge of simulating a set of 3 deer objects and 2 lion objects. The Simulation steps through 20 turns. In each turn, each Deer and Lion object moves a random direction (north, south, east, or west). When all moves for each organism have been made (in a single turn), the Ecosystem checks the location of each Lion and Deer to see if any Lion is also in the same space as a Deer. If a Lion is in the same space as a Deer, the Ecosystem has that Lion call its eat(Deer) method on the deer that is also occupying that coordinate. The Lion can only eat one Deer per turn. The process happens for each of the 20 steps of the simulation.