```
Class Transportation
        public:
        virtual bool crossTerrain(Terrain& t) =0; // pure virtual to be implemented by child classes
        Void updateColor(string& newcolor); // Allows player to update color of their
transportation

        private:
        string color;

Class Horse public : Transportation
        public:
        bool crossTerrain(Terrain& t) {
                if (t is crossable by horse) { return true; }
                else { return false; }
                }
        private:
        string color;

Class Ship : public Transportation
        public:
        bool crossTerrain(Terrain& t) {
                if (t is crossable by ship) { return true; }
                else { return false; }
                }
        private:
        string color;

Class Airship public : Transportation
        public:
        bool crossTerrain(Terrain& t) {
                if (t is crossable by airship) { return true; }
                else { return false; }
                }
        private:
        string color;
```

**** the four different types of terrain would be passed into each of the crossTerrain(Terrain& t) functions that each type of Transportation has a separate implementation of.**

```
Class Player
        public:
        void attack() { curr->useWeapon(); }
        void setWeapon(Weapon& type) { search vector for Weapon type and set curr to that
type of weapon if the player owns it }
        … could also have these for armour and shelter

        private:
        vector<Transportation*> t;
        vector<Weapon*>;
        vector<Armour*>;
        vector<Shelter*>;
        Weapon* curr; // Current weapon in use
        Int strength;
        Int experience;
```

```
Class Potion
        public:
        virtual void drinkPotion() = 0;
```

The following inherit from the Potion Class:

```
Class HealingPotion : public Potion
        void drinkPotion() { strength++; }

Class MagicPotion : public Potion {
        void drinkPotion() { experience++; }
```

```
Class Weapon
        public:
        virtual void useWeapon()=0;
```

The following inherit from the Weapon Class:

```
Class Sword : public Weapon
        void useWeapon();

Class Spear : public Weapon
        void useWeapon();
```

```
Class Crossbow : public Weapon
        void useWeapon();
```

Each derived class will provide its own implementation of useWeapon();

```
Class Armour
        public :
        virtual void useArmour();
```

The following inherit from the Armour Class:

```
Class Shield : public Armour
        void useArmour();

Class Helmet : public Armour
        void useArmour();

Class ChainMail : public Armour
        void useArmour();
```

Each derived class will provide its own implementation of useArmour();

```
Class Shelter
        public:
        void useShelter() = 0;

Class Tent : public Shelter
        Public:
        void useShelter();
```