Douglas Sangster – Implementation & Testing Unit (PDA)

- I.T. 1 – Use of Encapsulation

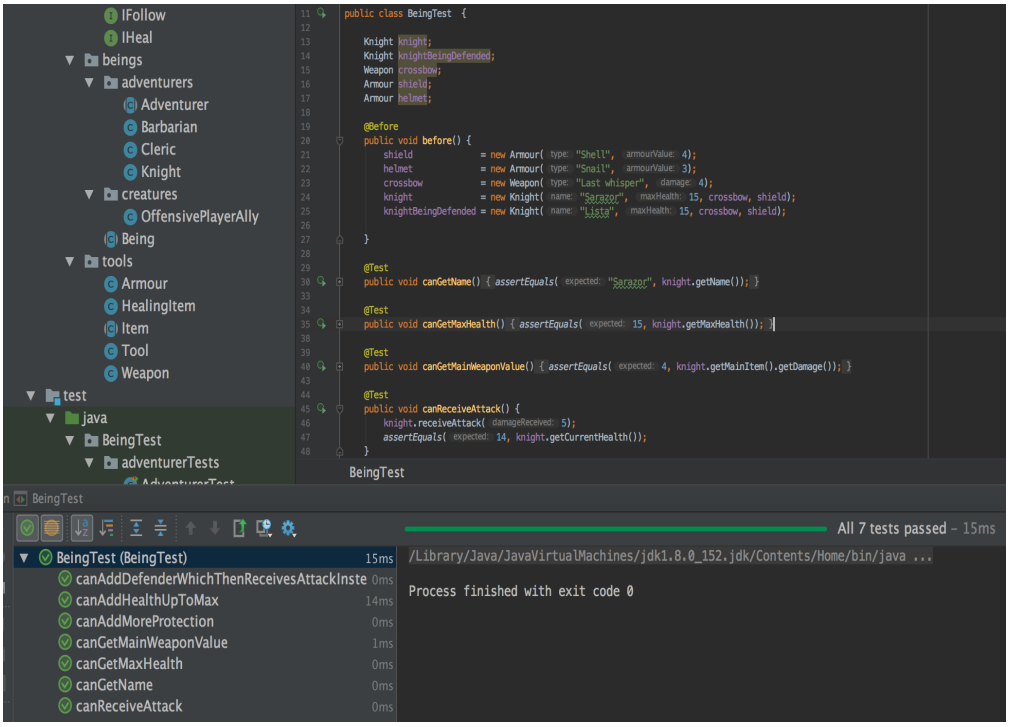
```java
public class Card implements Serializable {


    private CardName        name;
    private FactionColour   colour;
    private int             imageBlueViewInt;
    private int             imageRedViewInt;
    private HashMap<Integer, ArrayList<Coordinate>> moveset;


    public Card(CardName name, FactionColour colour, int imageBlueViewInt, int imageRedViewInt, HashMap<Integer, ArrayList<Coordinate>> moveset) {

        this.name               = name;
        this.colour             = colour;
        this.imageBlueViewInt   = imageBlueViewInt;
        this.imageRedViewInt    = imageRedViewInt;
        this.moveset            = moveset;
    }

    public CardName getName() {

        return name;
    }

    public FactionColour getCardColour() {

        return this.colour;
    }

    public int getImageBlueViewInt(){

        return this.imageBlueViewInt;
    }

    public int getImageRedViewInt(){

        return this.imageRedViewInt;
    }
```

- I.T. 2 – Use of Inheritance

```java
public abstract class Being {

    private String name;
    private int maxHealth;
    private int currentHealth;
    protected IDefend defender;
    protected ArrayList<IAbsorb> protection;


    public Being(String name, int maxHealth) {
        this.name           = name;
        this.maxHealth      = maxHealth;
        this.currentHealth  = maxHealth;
        this.protection     = new ArrayList<>();
    }


    public void receiveAttack(int damageReceived){
        if(defender != null){
            ((Being) defender).receiveAttack(damageReceived);
        } else {
            for(IAbsorb element: protection){
                damageReceived = element.block(damageReceived);
            }
            takedamage(damageReceived);
        }
    }

    public String getName() { return name; }

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

    public int getCurrentHealth() { return this.currentHealth; }
```

```java
public abstract class Adventurer<T> extends Being {


    protected T mainItem;

    public Adventurer (String name, int maxHealth, T mainItem) {
        super(name, maxHealth);
        this.mainItem = mainItem;
    }

    public abstract void useMainItem(Being target);

    public T getMainItem(){
        return this.mainItem;
    }

    public void wieldItem(T newMainItem) {
        this.mainItem = newMainItem;
    }
}
```

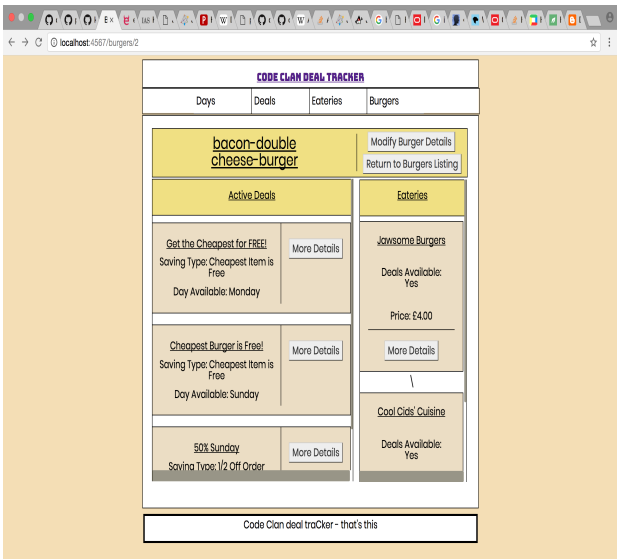
```java
public class Knight extends Adventurer<Weapon> implements IDefend {


    private Being beingUnderProtection;


    public Knight(String name, int maxHealth, Weapon mainWeapon, Armour shield) {
        super(name, maxHealth, mainWeapon);
        this.protection.add(shield);
    }

    public void useMainItem(Being target) {
        this.mainItem.dealDamage(target);
    }

    public void defend(Being target){
        if(this.defender != null){
            this.defender.stopDefending();
        }

        if(beingUnderProtection != null) {
            beingUnderProtection.removeDefender();
        }

        beingUnderProtection = target;
        target.addDefender(this);
    }

    public void stopDefending(){
        beingUnderProtection.removeDefender();
        beingUnderProtection = null;
    }
```

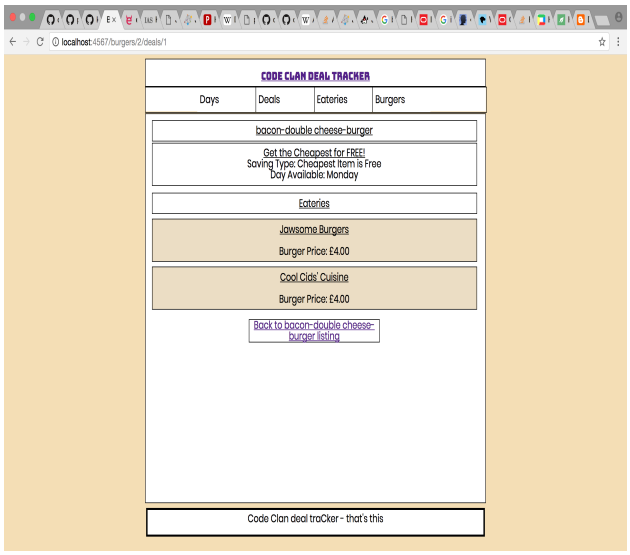


- I.T. 3 – Searching Data

```ruby
def Eatery.find_by_burger_deal(options)
  sql = "
  SELECT eateries.id, eateries.name
  FROM eateries INNER JOIN active_deals ON
  active_deals.eatery_id = eateries.id WHERE
  active_deals.burger_id = $1 AND
  active_deals.deal_id = $2;
  "
  burger_id = options['burger'].to_i
  deal_id = options['deal'].to_i
  values = [burger_id, deal_id]
  eatery_hashes = SqlRunner.run(sql, values)
  return mapper_aid(eatery_hashes)
end
```

```
get('/burgers/:burger_id/deals/:deal_id')
do
  burger_id = params['burger_id'].to_i
  deal_id = params['deal_id'].to_i
  @burger = Burger.find(burger_id)
  @deal = Deal.find(deal_id)
  @eateries =
  Eatery.find_by_burger_deal({'burger' =>
  burger_id , 'deal' => deal_id})
  erb(:"burgers/deals/show")
end
```

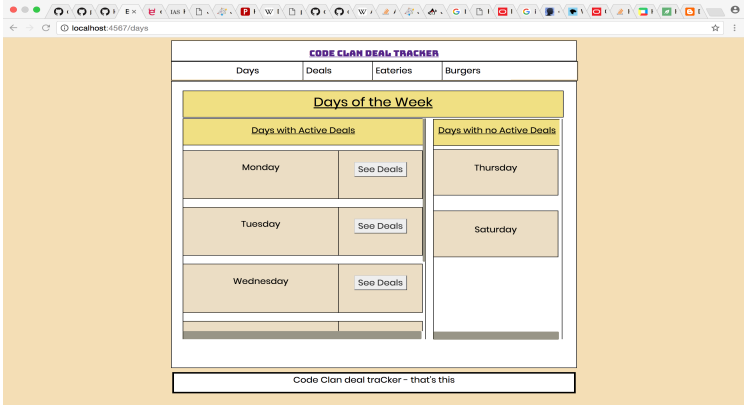




- I.T. 4 – Sorting Data

```
def Day.find_all_active
  sql = "
  SELECT DISTINCT days.id, days.day FROM
  days INNER JOIN
  deals ON days.id = deals.day_id INNER
  JOIN
  active_deals ON deals.id =
  active_deals.deal_id
  ORDER BY days.id ASC;
  "
  day_hashes = SqlRunner.run(sql)
  return mapper_aid(day_hashes)
end
```

```
get('/days') do
  @active_days = Day.find_all_active
  @inactive_days = Day.find_all_inactive
  erb(:"days/index")
end
```



- I.T. 5 – Use of an Array

```
public Deck(){
    cards = new ArrayList<>();
    createDeck();
}

private void createDeck() {
    Suit[] suits = Suit.values();
    Rank[] ranks = Rank.values();
    int indexSuit;
    int indexRank;
    for (indexSuit = 0; indexSuit < suits.length; indexSuit += 1) {
        for (indexRank = 0; indexRank < ranks.length; indexRank += 1) {
            Card card = new Card(suits[indexSuit], ranks[indexRank]);
            cards.add(card);
        }

    }
}


public int getSize() {
    return this.cards.size();
}
```

```
@Before
public void before(){
    deck = new Deck();  deck: Deck@832
    player = new Player( name: "Colin");  player: null
    dealer = new Dealer( name: "Joker");
}

@Test
public void deckHas52Cards(){
    assertEquals( expected: 52, deck.getSize());
}
```

DeckTest › before()

riables

this = {DeckTest@827}
deck = {Deck@832}
cards = {ArrayList@841} size = 52

- I.T. 6 – Use of a Hash

```
public HashMap<Player,GameResultType> decideResult(Player player1, Player player2) {
    HashMap<Player, GameResultType> outcome;
    outcome = new HashMap<>();
    if (player1.getHandValue() == player2.getHandValue()){
        outcome.put(player1, GameResultType.DRAW);
        outcome.put(player2, GameResultType.DRAW);
    } else if (player1.getHandValue() > player2.getHandValue()){
        outcome.put(player1, GameResultType.WIN);
        outcome.put(player2, GameResultType.LOSE);
    } else {
        outcome.put(player1, GameResultType.LOSE);
        outcome.put(player2, GameResultType.WIN);
    }

    return outcome;
}
```

```
    adjudicationResultHash = adjudicator.decideResult(this.playerHuman, this.playerComputer);
    GameResultType playerHumanResult = adjudicationResultHash.get(this.playerHuman);

    if (playerHumanResult.equals(GameResultType.DRAW)){
        resultString = "The game is a draw!";
    } else if (playerHumanResult.equals(GameResultType.WIN)) {
        resultString = "You won the game!";
    } else {
        resultString = String.format("You lost the game, %s had a better hand.", opponentName);
    }

    System.out.println(resultString);
}
```

```
classes — java Runner — java — java — 80×24
Your hand has a value of 19, Alex's hand has a value of 11.

You won the game!


Start another game?
(Y)es/(N)o:
```

- I.T 7 – Use of Polymorphism

```java
public class AntiAir extends Vehicle {

    private ArrayList<IFly> targetsLocked;


    public AntiAir(String model, int healthValue, int attackValue, int accuracyValue) {
        super(model, healthValue, attackValue, accuracyValue);
        targetsLocked = new ArrayList<>();
    }

    public void lockOnTarget(IFly target){
        targetsLocked.add(target);
    }

    public ArrayList<IFly> getTargetsLocked(){
        return targetsLocked;
    }
}
```

```java
public class FantasyKaiju extends Kaiju implements IFly {


    public FantasyKaiju(String name, int healthValue, int attackValue, int accuracyValue, EntitySize size){

        super(name, healthValue, attackValue, accuracyValue, size);

    }


    public String roar() { return "Sparks fly everywhere as a heavy roar rushes by"; }

    public String fly(){
        return "You hear the flapping of wings as you see a " + this.name + " flying through the air";
    }

}
```

```java
public class JetFighter extends Vehicle implements IFly {


    public JetFighter(String model, int healthValue, int attackValue, int accuracyValue){

        super(model, healthValue, attackValue, accuracyValue);

    }


    public String fly(){
        return String.format("%s zooms through the air", this.model);
    }


}
```

```java
public interface IFly {

    String fly();

}
```